Introduzione a JavaScript - Parte 1

ECMAScript

- Linguaggio di script client-side, standardizzato da ECMA International.
- Nato nel 1995 come JavaScript (Sun e Netscape).
- Standardizzato nel 1997 (ECMA).
- Versione 6 (ES6/EcmaScript 2015): importanti cambiamenti e nuovi costrutti.
- Versione 14 (ES14/EcmaScript2023): Versione attuale.
- ES.Next: nome dinamico per le feature in sviluppo.
- Approccio "living standard" del WHATWG.

Esecuzione di script JavaScript

- **Client-side (eventi):**
  - Attributi `on+evento` degli elementi HTML (es. `onclick`, `onmouseover`).
  - Callback a funzioni JavaScript.
  - Eventi speciali: `load` (window) e `ready` (document).
- **Server-side (routing):**
  - URI associati a servizi (es. file separati).
  - Node.js (Express.js): callback JavaScript associate a URI.
- **Modalità di esecuzione nel browser:**
  - **Sincrona:** `<script>` o file esterno, esecuzione immediata.
  - **Asincrona (eventi):** codice associato a eventi del documento.
  - **Asincrona (Ajax):** callback eseguite al completamento di richieste HTTP.
  - **Asincrona (timeout):** esecuzione dopo un periodo di attesa.
  - **Nota:** L'esecuzione di script blocca il browser.

Output degli script

- **`document.write(string)`:** Scrive direttamente nella finestra del browser.
- **`console.log(string)`:** Scrive sulla console del browser.
- **`alert(string)`:** Mostra una finestra di alert.
- **Modifica del DOM:** `document.getElementById(id).innerHTML = string;`

Attivazione degli script in HTML

1. **Attributo di un evento:** `<button onclick="myFunction()">`
2. **Tag `<script>`:** `<script> /* codice JavaScript */ </script>`
3. **File esterno:** `<script src="myScript.js"></script>`

Tipi di dato

- **Atomici (built-in):**
  - Booleani (`true`, `false`)
  - Numeri (interi e floating point)
  - Stringhe (racchiuse tra apici singoli o doppi)
  - `null`
  - `undefined`
- **Strutturato:**

    - `object` (include gli array)

Variabili

- **Tipizzazione dinamica:** le variabili non hanno un tipo fisso.
- **Dichiarazione:**
    - `var` : scope della funzione o del file.
    - `let` : scope del blocco o della riga.
    - `const` : variabile costante (non modificabile).

Operatori

- **Numeri:**
    - `+` (somma)
    - `-` (sottrazione)
    - `*` (moltiplicazione)
    - `/` (divisione)
    - `%` (modulo)
    - `**` (esponente)
    - `++` (incremento)
    - `--` (decremento)
- **Stringhe:**
    - `+` (concatenazione)
    - Concatenazione con casting (es. `"5" + 7` risulta `"57"`)
- **Confronto e booleani:**
    - `==` (uguaglianza con casting)
    - `===` (uguaglianza senza casting)
    - `!=` (disuguaglianza con casting)
    - `!==` (disuguaglianza senza casting)
    - `<` (minore)
    - `>` (maggiore)
    - `<=` (minore o uguale)
    - `>=` (maggiore o uguale)
    - `&&` (AND)
    - `||` (OR)
    - `!` (NOT)

Strutture di controllo

- **Condizionali:**
    - `if` / `else`
    - Operatore ternario: `(condizione ? valoreSeVero : valoreSeFalso)`
    - `switch` / `case` / `break` / `default`
- **Cicli:**
    - `for`
    - `for...in` (itera sulle proprietà di un oggetto)
    - `while`
    - `do...while`
- **Eccezioni:**

    - `try` / `catch` (gestione degli errori)
    - Programmazione "paranoica" (controlli manuali) vs. `try...catch`.

Funzioni

- Blocchi di istruzioni con nome e parametri (opzionali).
- Possono restituire un valore (`return`).
- Non tipizzate (i valori di ritorno sono tipati).
- Parametri mancanti: assumono il valore `undefined`.
- Funzione semplificata (arrow function) `function double(n) { return n? n+n : 0;}`

Tipi di dati strutturati (Oggetti)

- Liste non ordinate di proprietà (coppie nome-valore).
- Valori delle proprietà: possono essere altri oggetti (annidamento).
- **Sintassi di accesso:**
    - Dot syntax: `object.property`
    - Square bracket syntax: `object['property']`
    - La square bracket syntax permette di usare variabili e espressioni per il nome della proprietà.
- Lettura e scrittura delle proprietà con entrambe le sintassi.

Array

- `object` speciali con chiavi numeriche intere (assegnate automaticamente).
- Dichiarazione: parentesi quadre (`[]`).
- Accesso: solo square bracket syntax (es. `array[0]`).
- **Proprietà e metodi utili:**
    - `length` : lunghezza dell'array.
    - `indexOf(item)` : posizione di un elemento.
    - `pop()` : rimuove e restituisce l'ultimo elemento.
    - `push(item)` : aggiunge un elemento alla fine.
    - `shift()` : rimuove e restituisce il primo elemento.
    - `unshift(item)` : aggiunge un elemento all'inizio.
    - `slice(start, end)` : restituisce una porzione dell'array.
    - `splice(pos, rimuovi, inserisci)` : inserisce e rimuove elementi.
    - `join(sep)` : crea una stringa da un array.
- Annidamento di oggetti e array.

Oggetti predefiniti

- **Multipli:**
    - `Object`
    - `Array`
    - `String`
    - `Date`
    - `Number`
    - `RegExp`

    - ...
- **Singoletti:**
    - `Math`
    - `JSON`
    - ...

Stringhe

- Metodi dell'oggetto `String`:
    - `length`: lunghezza.
    - `indexOf(sub)`: posizione di una sottostringa.
    - `substring(start, end)`: sottostringa da `start` a `end`.
    - `substr(start, length)`: sottostringa da `start` per `length` caratteri.
    - `split(sep)`: divide una stringa in un array.

JSON (JavaScript Object Notation)

- Formato dati derivato dalla notazione degli oggetti JavaScript.
- **Regole:**
    - Valori: stringhe, numeri, booleani, array, oggetti.
    - Nomi delle proprietà tra virgolette doppie.
    - Solo virgolette doppie.
    - Nessun commento.
- **Metodi del singoletto `JSON`:**
    - `JSON.stringify(object)`: converte un oggetto in una stringa JSON.
        - Si può aggiungere come secondo parametro null, e come terzo parametro un numero per avere l'indentazione a `n` spazi.
    - `JSON.parse(string)`: converte una stringa JSON in un oggetto.

Date

- Rappresentazione: numero di millisecondi dal 1 gennaio 1970.
- **Costruttore:** `new Date()` (data e ora correnti), `new Date(anno, mese, giorno)` (mesi da 0 a 11).
- **Metodi:**
    - `getDay()`: giorno della settimana (0-6).
    - `toDateString()`: converte in stringa leggibile.
    - `toLocaleDateString()`: versione localizzata
    - Operazioni aritmetiche e confronti possibili (perché è un numero).

Altri oggetti

- **`Math`:**
    - Singoletto con funzioni e costanti matematiche (es. `Math.PI`, `Math.sin()`, `Math.random()`).
- **`RegExp`:**
    - Espressioni regolari per il pattern matching su stringhe.
    - Delimitatore: `/`.
    - Esempio: `let re = /pi(.)/; let x = str.match(re);`