Modern JavaScript

OBIETTIVI

- L'obiettivo di questo corso è quello di conoscere gli strumenti comunemente usati nello sviluppo React, in modo che possiate più agevolmente seguire i successivi corsi in programma.
- Lo sviluppo su React richiede una buona conoscenza di JavaScript, inoltre viene fatto ampio uso di una sintassi aggiornata. In questo corso impareremo a conoscere la moderna sintassi JavaScript.
- Non verranno fornite nozioni architetturali, ad esempio su come strutturare una applicazione front-end o su strutturare una back-end Node.js.
- Ci focalizzeremo sulla sintassi JavaScript considerata ufficialmente parte del linguaggio al momento di questo corso, con maggiore attenzione a quelle caratteristiche che incontriamo quotidianamente nello sviluppo React.
- Sebbene non affronteremo caratteristiche sperimentali o linguaggi differenti (come TypeScript), le nozioni che cercheremo di trasmettervi vi permetteranno di esplorare questi fronti in autonomia.

LA COMPILAZIONE NEL MONDO JS

- Se siamo abituati a JavaScript senza step di compilazione, negli ultimi anni molti fattori hanno concorso a stabilizzare la pratica di introdurre uno step di compilazione nei processi di build delle applicazioni front-end web.

NASCITA DEI TRANSPILER

- Intorno al 2010 nasce e si diffonde CoffeeScript, linguaggio che aggiunge zucchero sintattico a JavaScript con una sintassi ispirata a Ruby.
    - Viene compilato con compilatore apposito su Node.
- Negli stessi anni (2010) viene rilasciato da Google *Closure Compiler* che si occupa di analizzare, ottimizzare e minificare codice JavaScript e di "compilare" verso ES3
- Nascita di React da spinta a utilizzo di transpiler dato l'utilizzo di JSX (che non è una feature ufficiale di JavaScript)
    - Sviluppato da Facebook, utilizzato da prima su NewsFeed FB e poi su Instagram e reso Open Source dal 2013

NASCITA DEI TRANSPILER

- Principale transpiler che viene utilizzato al giorno d'oggi è Babel, nato con il nome di "6to5" nel 2014 si occupava della conversione di ES6 in ES5.
    - Con i transpiler abbiamo la possibilità di rendere retro-compatibile JavaScript moderno su browser meno aggiornati.
- Le garanzie di stabilità e qualità del codice offerte da Typescript spingono nuovamente la community ad accogliere i transpiler.

BUNDLER

- Nel 2010 il registro npm (originariamente pensato per Node.js) fa gola agli sviluppatori JavaScript front-end web. Nel 2011 viene rilasciato Browserify che permette di usare "require" nel mondo web.
- Browserify include da subito il supporto per i transpiler (in particolare per CoffeeScript)
- Browserify si poneva come obiettivo l'aggregazione di sorgente recuperato tramite package manager di node in un unico "pacchetto" (bundle) pronto per l'esecuzione su un browser, quindi traslare le dipendenze della propria runtime Node su runtime del browser.
- Questo rimpiazza la tradizionale modalità di concatenazione sequenziale che ha regnato fino a quel momento.
- Browserify è ora stato superato da strumenti più moderni: Webpack, Parcel e Rollup per citarne alcuni.

MOTIVAZIONI

- Perchè sviluppare front end in questo modo?
- Compatibilità browser, ovvero si può sviluppare in maniera agnostica rispetto al browser di destinazione, demandando prefixing e polyfill al processo di build. (Stessa cosa avviene nel CSS)
    - Prefixing - utilizzo di regole CSS specifiche per differenti browser
    - Polyfill - codice aggiuntivo che si occupa di fornire servizi o funzionalità non presenti su un browser specifico
- Possibilità di utilizzare feature nuove di JavaScript e non presenti su vecchi browser.
- Possibilità di utilizzare linguaggi di programmazione differenti.
- Gestione automatizzata (no download + unzip + move, no copy-pasta), semplificata (un solo comando) e strutturata (ovvero "esplicita") delle dipendenze
- Modularizzazione e riutilizzo strutturato del codice