UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

APPUNTI DI "Applicazioni dinamiche per il web"

Laurea Magistrale in *Ingegneria e scienze informatiche*

A.A. 2020 – 2021

A cura di *Davide Zampieri*

Basati sulle slide fornite dal *prof. Roberto Posenato*

INTRODUZIONE ALLE TECNOLOGIE DEL WORLD WIDE WEB

Il *World Wide Web Consortium* (W3C) ha diretto lo sviluppo dei più importanti standard del Web: URI, HTTP, CSS, XML (e standard collegati) e accessibilità sul web.
Il *Web Hypertext Application Technology* (WHAT) propose delle modifiche a HTML, CSS, DOM e JavaScript che cambiarono alcuni aspetti di stretta competenza del W3C, il quale dovette ammettere l'impatto innegabile di queste modifiche: nasce HTML 5.

Il *World Wide Web* (WWW) è un sistema per la presentazione a schermo di documenti multimediali e per l'utilizzo di link ipertestuali per la navigazione.
Il *client* (o browser) è un visualizzatore di documenti ipertestuali e multimediali. Può visualizzare testi, immagini e semplici interfacce grafiche. I browser hanno anche dei plug-in che permettono di visualizzare ogni tipo di formato speciale e un linguaggio di programmazione interno (JavaScript) che permette di realizzare applicazioni autosufficienti.
Il *server* è un meccanismo di accesso a risorse locali (file, record di database, …) ed è in grado di trasmetterle via socket TCP. Il server può collegarsi ad applicazioni server-side ed agire da tramite tra il browser e l'applicazione in modo che il browser diventi l'interfaccia dell'applicazione.

Alla base del WWW ci sono i seguenti *protocolli*:

- *URI*, ovvero uno standard per identificare in maniera generale le risorse di rete e per poterle specificare all'interno di documenti ipertestuali.
- *HTTP*, ovvero un protocollo di comunicazione state-less e client-server per l'accesso a risorse ipertestuali via rete.
- *HTML*, ovvero un linguaggio per la realizzazione di documenti ipertestuali basato su SGML e incentrato sulla possibilità di realizzare connessioni ipertestuali nella descrizione strutturale del documento.

Il *linguaggio HTML* non nacque per realizzare applicazioni interattive sofisticate attraverso siti web. Per questo motivo sono state create tecnologie aggiuntive per ottenere questi effetti; le più importanti sono: JavaScript, CSS e DOM.

REpresentational State Transfer (REST) è un modello di interazione tra client e server basato sul protocollo HTTP. Una corretta applicazione di REST garantisce interoperabilità, scalabilità e uso delle caratteristiche più avanzate di HTTP.

Responsive web design: la grande varietà di dispositivi obbliga a realizzare pagine che si adattino ad essi; una pagina responsive si adatta elegantemente a qualunque tipo di schermo al meglio delle sue possibilità.

I *framework* sono librerie che semplificano radicalmente la progettazione di pagine e servizi web; inoltre, uniformano le differenze dei singoli browser. I framework possono essere più (opinionated) o meno rigidi e restrittivi (non-opinionated) nell'imporre un modello di design specifico.

Una pagina web richiede la coordinazione di parti visuali, stile, computazione, eccetera. Nel *component-based design* queste parti vengono sviluppate in maniera integrata per ogni componente. Quindi un componente contiene frammenti di HTML (spesso chiamati template), classi CSS e codice JavaScript, complessivamente autosufficienti e necessari per il buon funzionamento del componente. Il component-based design è stato introdotto dal framework Angular ed è diventato estremamente popolare negli ultimi anni, contribuendo alla creazione di altri framework (tra cui React e Vue).

Principi architetturali del WWW:

- *Identificazione* – nel WWW ogni elemento di interesse è chiamato risorsa ed è identificato da un identificatore globale chiamato Uniform Resource Identifier (URI).
- *Interazione* – il protocollo di comunicazione HTTP permette di scambiare messaggi su una rete TCP/IP.
- *Formato* – il contenuto di una risorsa deve avere un formato di dati definito e chiaro in modo da permettere di accedere al suo contenuto in maniera chiara e non ambigua (ad es. MIME).

MIME permette di:

- Scrivere messaggi di testo usando diversi set di caratteri.
- Avere un insieme estensibile di formati per messaggi non testuali.
- Avere messaggi multi-parte.

Tutti i messaggi MIME vengono identificati da un *Content Type* che definisce il tipo di dati del messaggio e aiuta l'applicazione ricevente a gestire il messaggio e a invocare l'applicazione più adatta. Inoltre, un messaggio MIME può contenere parti di tipo diverso: in questo caso si creano dei sotto-messaggi MIME per ciascuna parte e il messaggio MIME complessivo diventa multi-parte, qualificando (Content-Type) e codificando (Content-Encoding) in maniera diversa ognuna delle sue sotto-parti.

APPROFONDIMENTI SU HTTP E TECNOLOGIA REST

Le *caratteristiche di base* di HTTP sono:

- È *semplice*, perché è un protocollo testuale (leggibile).
- È *estensibile*, perché le intestazioni (headers) sono parti del protocollo che permettono di estendere le capacità del protocollo stesso.
- È basato su schema *client-server*, in quanto il client attiva la connessione e richiede dei servizi mentre il server accetta la connessione, nel caso identifica il richiedente, risponde alla richiesta e alla fine chiude la connessione.
- È un *protocollo generico*, in quanto è indipendente dal formato dati con cui vengono trasmesse le risorse (ad es. documenti HTML, file binari, eseguibili, oggetti distribuiti, altre strutture dati più o meno complicate).
- È *stateless* ma non sessionless, in quanto il server considera due richieste successive (anche sulla stessa connessione) come indipendenti (stateless) ma tramite i cookies è possibile creare delle sessioni (insiemi di richieste collegate fra loro che possono condividere dati sul server).

HTTP può *controllare*:

- Il *caching*, in quanto permette di indicare come un documento può essere mantenuto valido (cached) e se ignorare documenti cached.
- L'*origine dei documenti*, in quanto un client permette l'accesso ai dati recuperati da una sorgente solo a script provenienti dalla stessa sorgente (gli HTTP headers possono estendere l'origine di un documento permettendo la composizione di documenti da diverse sorgenti).
- L'*autenticazione*, in quanto sorgenti di dati possono essere protette in modo che solo gli utenti autenticati possano accedervi.
- *Proxy* e *tunneling*, in quanto server e client possono essere in zone di Internet non accessibili direttamente (firewall).
- Le *sessioni*, gestendo i cookies (una sessione può essere stabilita e distrutta a livello di protocollo).

I metodi di una *richiesta* HTTP sono:

- *GET*, richiesta di (una copia della rappresentazione di) una risorsa; è idempotente.
- *HEAD*, richiesta di (una copia della rappresentazione di) meta-informazioni di una risorsa per valutare caratteristiche della risorsa stessa; è idempotente.
- *POST*, richiesta di associare delle informazioni ad una risorsa; non è idempotente.
- *PUT*, richiesta di inserire (creando o sostituendo) una risorsa; è idempotente.
- *DELETE*, richiesta di rimozione di una risorsa e di ogni sua meta-informazione; è idempotente.
- *OPTIONS*, richiesta dell'elenco delle opzioni attive sul server web (serve anche per testare la raggiungibilità di un sito).

La classi di una *risposta* HTTP (status code) sono:

- *1xx – Informational*, una risposta temporanea alla richiesta (durante il suo svolgimento).
- *2xx – Successful*, il server ha ricevuto, capito e accettato la richiesta.
- *3xx – Redirection*, il server ha ricevuto e capito la richiesta ma sono necessarie altre azioni da parte del client per portare a termine la richiesta.
- *4xx – Client error*, la richiesta del client non può essere soddisfatta per un errore da parte del client (errore sintattico o richiesta non autorizzata).
- *5xx – Server error*, la richiesta può anche essere corretta ma il server non è in grado di soddisfare la richiesta per un problema interno.

Per contrastare la *cross-site vulnerability*, i browser rifiutano qualunque connessione JavaScript ad un dominio diverso da quello della pagina ospitante (dominio diverso = schema o dominio o porta diversa). Quindi solo gli script che hanno origine nello stesso dominio della pagina HTML verranno eseguiti.
Una soluzione a questa politica conservatrice è rappresentata dal *Cross-Origin Resource Sharing* (CORS), il quale prevede l'uso di due nuovi header: Origin (nella richiesta) per specificare il dominio su cui si trova il contenuto; Access-Control-Allow-Origin (nella risposta) per indicare gli altri domini da cui è permesso caricare contenuti.

Un *web service* realizza un'interfaccia HTTP tra le funzioni di un'applicazione e applicazioni remote che vogliono usare tali funzioni. Le applicazioni remote possono essere: applicazioni automatiche che utilizzano i dati dell'applicazione; applicazioni web che mostrano all'utente menu di opzioni e form che gli permettono di eseguire un'azione sui dati dell'applicazione.
Le applicazioni server mettono a disposizione un'interfaccia detta *API* per essere usate dalle applicazioni client. Un'*applicazione REST* si basa sull'uso del protocollo HTTP e del naming (URI) per generare interfacce generiche di interazione con le applicazioni client. L'*architettura REST* si basa quindi su quattro presupposti:

1. Definire risorsa ogni concetto rilevante dell'applicazione web.
2. Associare ad ogni risorsa un URI (nome).
3. Esprimere ogni operazione dell'applicazione web come metodo HTTP:
    - Create (creazione di un nuovo oggetto) → metodo PUT.
    - Read (visualizzazione dello stato della risorsa) → metodo GET.
    - Update (cambio di stato della risorsa) → metodo POST.
    - Delete (cancellazione di una risorsa) → metodo DELETE.
4. Scambiare dati che siano una rappresentazione di uno stato della risorsa (il formato è specificato attraverso un Content-Type MIME).

REST prescrive di fornire URI a *individui* (intrinsecamente singolari) e *collezioni* (intrinsecamente plurali). Ogni operazione avviene su uno e uno solo di questi concetti. Il corpo di una richiesta e/o risposta non è la risorsa, bensì una rappresentazione della risorsa di cui gli attori si servono per portare a termine l'operazione.

LINGUAGGIO HTML

Con *HTML Living Standard* (HTML 5), il linguaggio di markup è stato rivisto globalmente, introducendo nuovi elementi più specifici per usi e abitudini ormai consolidate (ad es. nav, section, article invece di div e audio, video invece di object).
L'uso di script per la modifica dinamica in memory del documento (DOM) è diventata parte integrante del linguaggio e ne costituisce un elemento prevalente sulla sua linearizzazione in stringa HTML.

In HTML div e span sono detti *elementi generici*, in quanto sono privi di caratteristiche semantiche o presentazionali predefinite perché assumono quelle desiderate grazie ad attributi come style, class e lang; div mantiene la natura di elemento blocco, mentre span mantiene la natura di elemento inline (ogni altra caratteristica è neutra).

HTML 5 definisce anche gli elementi header e footer, i quali vengono usati sia per *contenuti ripetuti* (nel caso di paginazione) e sia per contenuti che solitamente sono visualizzati all'inizio o alla fine di una pagina.
Le *liste di navigazione* (elemento nav in combinazione con header/footer) sono molto utili per l'accessibilità, in quanto possono essere più facilmente identificate e accedute anche da utenti disabili (tramite screenreaders o altri ausili).

L'elemento *form* permette di acquisire dei valori stringa da tastiera da inviare a un server. Le form sono legate ad applicazioni server-side:

1. Il browser raccoglie dati dall'utente con una form.
2. Il browser crea una connessione HTTP con il server e invia tali dati.
3. Il server riceve i dati, li elabora e genera un documento di risposta che viene spedito al browser.

Gli *elementi* di una form sono: l'elemento form (il contenitore dei widget della form); gli elementi input e select (i widget per campi di inserimento dati e liste a scomparsa); l'elemento button (un bottone cliccabile); l'elemento label (l'etichetta per la parte visibile del widget).
In HTML 5 l'elemento *input* è arricchito con due attributi che permettono di controllare il focus e aggiungere suggerimenti agli utenti: placeholder (contiene una stringa che sarà visualizzata in un campo di testo se il focus non è su quel campo); autofocus (indica il campo su cui posizionare il focus al caricamento della form).

Il tag *meta* è un meccanismo generale per specificare meta-informazioni su un documento HTML, quali la codifica dei caratteri usata nel file e altre informazioni (keyword) utili ai motori di ricerca per l'indicizzazione. Inoltre, *<meta name="viewport">* aiuta a rendere pagine non progettate per il mobile leggibili anche su un dispositivo mobile, imponendo al dispositivo in questione (tipicamente uno smartphone) le caratteristiche da utilizzare per il viewport invece di quelle fisiche, in modo che le pagine non risultino illeggibili sui browser.

Il *Document Object Model* (DOM) è un'interfaccia di programmazione per documenti HTML che definisce la struttura logica dei documenti ed il modo in cui si accede e si manipola un documento. Utilizzando DOM i programmatori possono costruire documenti, navigare attraverso la loro struttura e leggere, aggiungere, modificare o cancellare elementi.

HTML prevede l'uso di *stili CSS* (Cascading Style Sheet) in quattro modi diversi: posizionati presso il tag di riferimento; posizionati nel tag style; importati dal tag style; indicati dal tag link. Inoltre, HTML permette di assegnare gli stili agli elementi in tre modi: assegnati a tutti gli elementi di un certo tipo (tramite il nome dell'elemento); assegnati a tutti gli elementi di una certa categoria (tramite il valore dell'attributo class); assegnati ad uno specifico elemento (tramite il valore dell'attributo id).

I *framework CSS* sono librerie già pronte, con regole associate ad elementi e classi, per essere utilizzate nei documenti HTML con poche o zero modifiche. Essi gestiscono le differenze di implementazione delle funzionalità CSS tra i vari browser, gestiscono layout responsive (che si adattano alla dimensione dello schermo) e forniscono un look integrato, omogeneo e professionale (tendendo però ad uniformare pesantemente il look dei siti web).
Bootstrap (il più famoso dei framework CSS) fornisce anche un lungo elenco di servizi grafici come tabulatori, navbar, finestre modali e pannelli di alta qualità grafica e funzionale.

JAVASCRIPT

Uno *script* JavaScript può essere eseguito:

- *Client-side* (eventi), ogni elemento del documento ha alcuni eventi associati e quindi inserendo istruzioni JavaScript (o chiamate a funzione) nel valore dell'attributo corrispondente all'evento si crea una chiamata callback.
- *Server-side* (routing), i servizi server-side sono associati a URI e quindi quando si apre una connessione a tali URI verrà poi invocato lo script ed eseguito il servizio (il modulo Express di NodeJS fornisce un meccanismo per associare una funzione/callback JavaScript ad ogni tipo di URI).

Durante l'esecuzione di uno script, il browser è bloccato e non reagisce agli input dell'utente.
HTML prevede l'uso di script in tre modi diversi: posizionati dentro all'attributo di un evento; posizionati nel tag script; indicati in file esterni puntati dal tag script.

I dati in JavaScript sono tipati mentre le variabili no. Inoltre, ci sono due modi per definire variabili: *var* definisce una variabile nello scope della funzione o del file in cui si trova; *let* definisce una variabile nello scope del blocco in cui si trova.

JavaScript Object Notation (JSON) è un formato dati derivato dalla notazione usata da JavaScript per gli oggetti. JSON è un singoletto in JavaScript che supporta due soli metodi: *stringify* e *parse*.

RegExp è la classe delle espressioni regolari, da usare per fare match su stringhe. Le RegExp usano il carattere '/' come delimitatore.

AJAX è un termine che indica l'uso di una combinazione di tecnologie comunemente utilizzate sul web allo scopo di garantire usabilità, velocità e portabilità. Tali tecnologie sono:

- HTML e CSS.
- DOM, modificato attraverso JavaScript per la manipolazione dinamica dei contenuti e dell'aspetto.
- XMLHttpRequest, per lo scambio di messaggi asincroni fra browser e web server.
- JSON, come meta-linguaggio dei dati scambiati.

Un'applicazione AJAX è divisa in tre momenti fondamentali:

1. Creazione e configurazione delle richieste per il server.
2. Esecuzione e memorizzazione delle risposte (o degli errori).
3. Modifiche al DOM della pagina.

I *framework* Ajax sono librerie JavaScript che semplificano la vita nella creazione di applicazioni Ajax anche complesse. I framework RIA (Rich Internet Application) come Angular, React e Vue includono una ricca collezione di widget, modelli di comunicazione client e server, funzionalità grafiche e interattive e spesso anche strumenti di sviluppo.

Peculiarità di JavaScript:

- JavaScript definisce come *falsy* quei valori che in caso di casting a booleano diventano falsi (ad es. 0, null, undefined).
- JavaScript definisce come *truthy* quei valori che in caso di casting a booleano diventano veri (inclusi "0", "null", "undefined").
- In JavaScript, le *funzioni* sono oggetti quasi come tutti gli altri e, infatti, possono essere assegnate a variabili, possono essere passate come parametri di funzione, possono essere restituite da una funzione, possono essere elementi di un oggetto, possono essere elementi di un array; inoltre, se non si fornisce un parametro ad una funzione che lo richiede, essa non restituisce errore ma assume che il parametro sia undefined.
- JavaScript è un linguaggio *object-oriented* (anche se non è tipato come Java) basato su prototipi invece di classi.
- Ogni oggetto in JavaScript è autonomo e si possono aggiungere tutti i metodi/proprietà che si vuole senza modificare gli altri, ma per aggiungere proprietà/metodi di oggetti creati in precedenza e che sono condivise da molti oggetti bisogna usare la proprietà *prototype*.
- JavaScript non ha protezione dei membri privati di un oggetto (sono tutti accessibili e manipolabili); lo scope *closure*, ossia lo scope della funzione all'interno della quale viene definita un'altra funzione (sempre accessibile alla funzione interna ma non dal mondo esterno), è il meccanismo usato per creare delle variabili interne veramente private.
- In EcmaScript 2015 (la standardizzazione presso ECMA di JavaScript) sono state aggiunte le *funzioni freccia*, ossia un nuovo modo per definire funzioni inline utile per definire semplici funzioni callback.
- In EcmaScript 2015 sono stati aggiunti anche i *template literals*, ossia una nuova sintassi per definire stringhe multi-linea con interpolazione di variabili.

L'interprete JavaScript prima esegue completamente lo script in corso e dopo esegue quelli in callback (programmazione asincrona). C'è necessità di *asincronicità* ogni volta che c'è l'esigenza di chiamare un servizio sulla cui disponibilità o sui cui tempi di esecuzione non c'è controllo. Tuttavia, il processo rimane bloccato in attesa del ritorno della funzione e quindi l'utente percepisce un'esecuzione a scatti non fluida (non responsive). Possibili soluzioni sono:

- Si usa una *funzione callback* che viene eseguita alla conclusione del servizio e che prende in input i risultati.
- Si usa una *promessa*, ossia un oggetto che rappresenta un valore che sarà determinato da un'operazione asincrona; una promessa permette di specificare che azioni eseguire quando l'operazione asincrona sarà conclusa (una funzione then per il caso in cui l'operazione è terminata con successo, una funzione catch per il caso di fallimento).
- Usare promesse semplificate tramite le istruzioni *async* (modificatore di funzione che dice che la funzione restituirà una promessa) e *await* (modificatore di promessa che sospende l'esecuzione fino a quando la promessa ritorna un valore).

ARCHITETTURE PER APPLICAZIONI WEB

Le *architetture a n-strati* hanno tutte le seguenti macro componenti (livelli):

- Presentazione (presentation).
- Logica (business logic).
- Dati (data).

Il numero n indica in quanti strati sovrapposti questi livelli sono organizzati.

In un'*architettura a 3-strati per web app* i vari livelli consistono in:

- *Presentation layer* (o front-end) – documenti statici o dinamicamente generati visualizzati da un browser.
- *Logic layer* (o middleware) – un'applicazione che processa le richieste e genera i documenti (ad es. NodeJS).
- *Data layer* (o back-end) – un database gestito da un software DBMS.

Nel pattern *Model-View-Controller* (MVC):

- *Model* – gestisce il comportamento e i dati dell'applicazione, risponde alle richieste di stato (solitamente dalla View), esegue istruzioni di cambio di stato (solitamente dal Controller).
- *View* – mostra i dati in una forma adatta per l'interazione (possono esistere più viste con differenti scopi per lo stesso insieme di dati).
- *Controller* – accetta input dall'utente e istruisce il Model e la View su quali azioni eseguire in funzione dell'input.

In pratica, per le *Web App* il MVC consiste in:

- Model – database (dati persistenti), mantenimento delle sessioni.
- View – HTML, CSS e template.
- Controller – codice JavaScript, processamento delle richieste HTTP.

Nel pattern *Model-View-ViewModel* (MVVM), ViewModel è un'astrazione della View che converte i dati del Model nel formato adatto per la View. La View interagisce con ViewModel via *data binding* e *richieste di nuove pagine*. Ciascuna vista ha un suo corrispondente nel ViewModel e un ViewModel può essere associato a più viste (gli stessi dati sono visualizzabili in modo diversi). Il successo di questo pattern sta nel fatto che la View è spesso realizzata usando componenti già pronti (ad es. Angular, React, Vue).

Nella *Component-Based Architecture* (CBA), una componente è un pezzo di codice che realizza un pezzo dell'interfaccia utente. Una componente è riusabile ed è indipendente dalle altre componenti presenti nella stessa pagina (ma può interagire con esse). Solitamente ogni componente è realizzata da una classe JavaScript (*Single Page Application* – SPA).
In CBA (ad es. React) le responsabilità sono divise orizzontalmente: template, logica e metodi per recuperare i dati di una componente sono tutti nella medesima classe nel livello View. In MVC, invece, le responsabilità sono divise verticalmente: template per UI, metodi per il controllo e metodi per recuperare dati sono in livelli differenti.

APPROFONDIMENTO SUI FRAMEWORK EXPRESS E REACT

Programmazione server-side (ad es. Express):

- Codice che si esegue sul server.
- Realizza la logica dell'applicazione (validazione dati e richieste, interrogazioni alla base di dati).
- In alcune architetture prepara anche HTML e CSS da inviare al browser.

Programmazione client-side (ad es. React):

- Codice che si esegue sul browser (JavaScript, HTML e CSS).
- Inizialmente era solo per migliorare la presentazione.
- In CBA serve anche per implementare parte della logica di applicazione.

Per facilitare lo sviluppo di applicazioni web lato server, solitamente si usano dei *framework*, ovvero un insieme di librerie che permette di sviluppare un'applicazione concentrandosi sulla logica dell'applicazione, facilitando e armonizzando tutte le tecnologie sottostanti necessarie a far funzionare un'applicazione web.

Express è un framework un-opinionated (non impone nessuna architettura per la web app) scritto in JavaScript e basato su Node.js, il quale è un JavaScript Runtime per gestire connessioni HTTP in modo veloce. Node.js è quindi un ambiente di esecuzione JavaScript che permette di: creare applicazioni server-side; installare e usare on-demand librerie di terze parti (tramite il package manager npm); gestire le connessioni con i client in maniera single-thread e non-blocking. Node.js è quasi esclusivamente basato su funzioni asincrone e callback. Inoltre, Node.js è estremamente modulare in quanto è possibile mettere codice indipendente in file JavaScript (moduli) ed eseguirlo secondo necessità.

Express fornisce una semplice interfaccia per fare *routing*

```
app.method(path, function(request, response) { ... })
```

dove:

- `method` è uno dei metodi HTTP.
- `path` è il local path dell'URI richiesto al server.
- `function(req, res)` è un handler da eseguire quando viene richiesto il path, in cui `req` è la richiesta HTTP (URI, intestazioni, parametri, dati) e `res` è la risposta HTTP in via di restituzione.

La libreria *body-parser* (integrata con Express) permette di accedere ai dati spediti dal POST nel body della richiesta convertendoli in oggetti accessibili dall'applicazione come: `<req>.body.<post_variable>`.
Express, quindi, ha pochissime funzionalità proprie, ma usa un grande numero di librerie *middleware* personalizzabili in uno stack di servizi progressivamente più complessi.
Si possono anche creare middleware personalizzati, per fare elaborazioni intermedie sulle request in entrata, facendo attenzione a specificare un terzo parametro (next) che il metodo middleware deve chiamare come ultima istruzione.

Vantaggi di Express:

- Ha un meccanismo di routing integrato in cui l'architettura REST è mappata bene.
- Offre potenzialmente molti servizi (ad es. interazione con database) ma non si è costretti a caricare tutte le librerie del framework per farlo funzionare in quanto è presente il package manager npm.
- Sfrutta i concetti di funzione asincrona e modulo tipici di NodeJS.

In un'applicazione web a 2-strati (CBA) la *logica sta sul client* e quindi:

- All'avvio dell'app viene caricata l'intera interfaccia utente HTML comprensiva di logica di applicazione nel browser.
- Le modifiche all'interfaccia, per via degli input dell'utente, vengono impostate tramite linguaggi di programmazione lato client (come JavaScript).
- Solitamente gli utenti lavorano per un periodo di tempo più lungo sulla stessa visualizzazione, perché solo i dati visualizzati sull'interfaccia utente vengono ricaricati dal server.
- Questo approccio viene usato prima di tutto per lo sviluppo di app Single-Page e viene facilitato da framework JavaScript.

Le *caratteristiche comuni* dei framework lato client sono:

- Accesso al DOM (modifiche dinamiche al contenuto degli elementi).
- Comunicazioni asincrone con il server (gestione successo ed errori, conversione da e per JSON).
- Gestione eventi (selezione dell'evento e associazione ad elementi arbitrari).
- Libreria di widget (collezione di elementi di interfaccia liberamente assemblabili per creare velocemente interfacce sofisticate e modulari).

I siti web dinamici creano le loro pagine HTML come risultato di un'operazione computazionale. Spesso si utilizza per questo un *template*, ovvero una pagina priva di contenuti specifici ma completa dal punto di vista grafico e tipografico, in cui vengono inseriti appositi elementi, detti placeholder, che verranno sostituiti con valori calcolati dall'applicazione per formare il documento completo. Nel tempo sono stati creati dei framework appositi per poter usare i template client-side.

React è un framework lato client che usa JavaScript come motore fondamentale, ma non usa HTML o il DOM come destinazione dei comandi, bensì View native del sistema operativo di destinazione. Non è opinioned né offre un sistema di routing esplicito (esistono comunque librerie esterne che lo permettono).
Un template in React è una funzione che restituisce codice HTML o *JSX*, ossia un'estensione JavaScript proposta da React che mescola in maniera sensata e intelligente codice JavaScript, frammenti HTML ed elementi XML inventati ad hoc (per i componenti).
Il metodo *render* specifica poi in quale frammento di DOM andare a posizionare il codice JSX interpretato (da Babel). Attenzione: solo quando si invoca render l'elemento viene aggiunto al DOM e quindi visualizzato, perché il *DOM virtuale* di React non è il DOM del documento.

In React un *componente* è un frammento di documento indipendente, riusabile e, generalmente, parametrizzato. Solitamente è realizzato da una funzione JavaScript che accetta parametri (chiamati props) e restituisce uno o più elementi JSX.
I componenti sono gli elementi di base che compongono una app. Ogni componente è modellato da una *classe* che controlla una parte dell'interfaccia utente chiamata View. Un componente può avere elementi dinamici ed elementi che rispondono ad eventi quali il click.
Inoltre, i componenti vengono creati, aggiornati e distrutti a seconda della navigazione dell'utente attraverso l'applicazione. Quest'ultima viene realizzata attraverso il *routing*: l'applicazione può passare da un macro-stato all'altro attraverso un meccanismo che modifica anche l'URI visibile e gestisce il passaggio dei dati da parte a parte dell'applicazione.

Vantaggi di React:

- Gestisce le differenze tra un browser e l'altro e fornisce un modello di programmazione e progetto dell'applicazione unico e omogeneo.
- È un ambiente di produzione di applicazioni web ispirato a CBA; è quindi basato sul concetto modulare di componente, che include anche i template HTML per la presentazione a schermo.
- Una Command Line Interface (CLI) permette di creare in modo automatico tutta la struttura (file system) e i file predefiniti per un'applicazione; si deve quindi solo modificare i file di interesse aggiungendo le componenti che rappresentano la logica dell'applicazione.
- È una libreria JavaScript che crea UI aggiornando un DOM virtuale con un flusso di dati unidirezionale.
- Quando il contenuto dinamico è predominante, l'aggiornamento e la gestione dei componenti è più semplice e veloce.
- È comodo se si vuole adottare l'approccio “tutto scritto in JavaScript” (ad es. assieme a Express).
- La documentazione è ben strutturata e le API di riferimento sono chiare.

Infine, è bene ricordare che la parte Model presente in CBA non è e non può essere il Model dell'applicazione. Un'applicazione web sicura, infatti, deve mantenere il *Model sul server* dove, dopo aver fatto la verifica di correttezza dei dati che arrivano dal client, si eseguono le operazioni di persistenza dei dati. Quindi i componenti di una web app che devono recuperare/salvare dati su base di dati devono poter accedere ai servizi di un Model che sta su un server.

ACCESSIBILITÀ

Per *accessibilità* si intende la possibilità che un sistema informatico (pagina web, applicazione) possa essere usato anche da persone con disabilità mediante tecnologie assistive. L'accessibilità non è vantaggiosa solo per le persone con disabilità, ma porta benefici anche a chiunque stia vivendo una disabilità temporanea o situazionale (derivante dal contesto). Bisogna preoccuparsi di questi aspetti non solo per motivi sociali ed etici, ma anche per ragioni legali e per motivi economici.

Le principali tecnologie assistive per la disabilità visiva sono gli screen reader e i video-ingranditori (screen magnifier). Uno *screen reader* è un'applicazione software che consente di comunicare le informazioni visualizzate su uno schermo con mezzi non visivi come la sintesi vocale, i suoni o un display Braille. Un software per l'*ingrandimento dello schermo*, invece, si interfaccia con l'output grafico di un computer per presentare una versione ingrandita del contenuto mostrato sullo schermo. Funzioni aggiuntive come l'inversione dei colori sono comunemente fornite per le persone con particolari difficoltà visive.

La maggior parte delle tecnologie assistive funziona emulando la tastiera. Ciò implica che è fondamentale che un sistema sia usabile esclusivamente mediante tastiera e navigabile usando il minor numero di tasti possibile. Infatti, la fatica può essere un problema quando sono necessarie molte sequenze di tasti per eseguire un compito, poiché l'utente potrebbe dover eseguire molti movimenti.

Web Content Accessibility Guidelines (WCAG 2.1) è una raccomandazione del W3C che contiene le linee guida che ogni sistema deve soddisfare per essere considerato accessibile. La conformità a WCAG 2.1 può essere classificata in tre diversi livelli (A, AA, AAA) a seconda dei criteri di successo che il sistema soddisfa:

- *A*, è il livello più basso di conformità; rimuove le principali barriere per cecità, sordità e disabilità motorie.
- *AA*, è il livello successivo di conformità (include A); rimuove le principali barriere per gli utenti ipovedenti e offre un piccolo aiuto per le disabilità cognitive.
- *AAA*, è il massimo livello di conformità (include A e AA); non è consigliabile che sia richiesto come politica generale per interi siti perché non è possibile soddisfare tutti i requisiti di livello AAA per alcuni contenuti.

Esempi pratici per garantire l'accessibilità sono:

- Codificare un sito usando HTML semanticamente valido; ad esempio, usare le rappresentazioni testuali equivalenti per le immagini e scrivere testi significativi per i link può aiutare gli utenti non vedenti.
- Se il testo e le immagini sono ingrandibili possono semplificare la lettura e la comprensione del contenuto da parte degli utenti con problemi di vista.
- Usare link differenziati (colorati, sottolineati) assicura che gli utenti daltonici possano notarli.
- Gestire correttamente il focus su link e aree cliccabili può aiutare gli utenti che non possono usare un mouse.
- In generale, scrivere codice che non ostacoli la navigazione mediante la sola tastiera.
- Se si scrivono i contenuti in un linguaggio semplice, essi risulteranno maggiormente comprensibili agli utenti con difficoltà di apprendimento.
- È molto più semplice rendere accessibile un'interfaccia facile da usare piuttosto che una complessa.

Accessible Rich Internet Applications (WAI-ARIA 1.1) è una raccomandazione del W3C che fornisce un'ontologia di ruoli, stati e proprietà che definiscono gli elementi dell'interfaccia utente e che possono essere utilizzati per migliorare l'accessibilità e l'interoperabilità dei contenuti e delle applicazioni Web.

È possibile usare WAI-ARIA sfruttando attributi specifici da applicare su qualsiasi elemento HTML:

- L'attributo *role* specifica il ruolo (semantica) dell'elemento (button, checkbox, tree, tablist, tab, …).
- Le proprietà come *aria-label* e *aria-labelledby* sono attributi che descrivono la natura di un dato oggetto.
- Gli stati come *aria-checked* e *aria-selected* sono proprietà dinamiche che esprimono le caratteristiche di un oggetto.

Non si deve usare ARIA quando non è necessario: se esiste un elemento HTML o un attributo nativo con la semantica ed il comportamento necessari bisogna usare quello.

Suggerimenti e trucchi:

- La navigazione di un'applicazione Web e l'interazione con essa tramite uno screen reader (ad es. NVDA, VoiceOver) offre una buona valutazione della sua accessibilità.
- Esaminare i problemi segnalati da strumenti di test di accessibilità automatizzati (ad es. Axe, Wave, Lighthouse).
- Se un elemento (ad es. il campo di una form) richiede l'uso di attributi particolari o l'uso della specifica WAI-ARIA per essere reso accessibile, bisogna riempire tali attributi con valori significativi.
- La gestione del focus è importante; un focus errato o mancante causa il disorientamento degli utenti di tecnologie assistive (ad es. una finestra di dialogo modale viene chiusa ma il focus non viene riportato sull'elemento che ne ha causato l'apertura).
- Se si usa un componente fornito da un framework esterno o un elemento HTML migliorato da esso, occorre sempre assicurarsi che sia accessibile o risolvere i suoi problemi di accessibilità; si consiglia di scegliere un framework noto per essere un buon punto di partenza in questo senso (ad es. Bootstrap v4).
- Prestare particolare attenzione al contrasto cromatico, alle dimensioni dei caratteri e a rendere il layout il più responsive possibile per rispondere alle modifiche delle dimensioni dei caratteri e allo zoom.