Appunti: Framework Javascript

Introduzione

Questo documento introduce diversi framework JavaScript a componenti, focalizzandosi su Angular.js, Angular, React.js, Vue.js e Web Components, con cenni ad altri framework emergenti.

Angular.js (versione 1.x)

- **Nascita e Storia:** Nasce nel 2009, progetto Open Source supportato da Google (2010).
- **Obiettivo:** Semplificare la creazione di Single Page Application (SPA).
- **Pattern:** Model-View-Controller (MVC) e Model-View-ViewModel (MVVM).
- **Concetto Chiave:** Trasformare parti di una pagina web in "view" di un "model", gestite da un "controller" Javascript (binding bi-direzionale).

Componenti Chiave di Angular.js

- **Model:**
  - Struttura dati (JSON o oggetto Javascript).
  - Può essere statico, caricato via Ajax, o calcolato.
- **View:**
  - Frammento HTML.
  - Decorato con attributi speciali (`ng-*`) e placeholder (`{{nome}}`).
  - Indica il controller da usare.
- **Controller:**
  - Codice Javascript.
  - Lega la view al model.
  - Utilizza la *dependency injection* (`$scope` , `$http`).
- **Module:**
  - Contenitore di tutte le parti dell'applicazione.
- **Directive:**
  - Struttura di markup (elemento, attributo, classe) con un comportamento specifico.

Angular (versione 2+)

- **Evoluzione Radicale:** Rilascio drasticamente differente da AngularJS (fine 2014), *non* retrocompatibile.
- **Nomenclatura:**
  - **AngularJS:** versione 1.x.
  - **Angular:** versioni successive (senza specificare la versione).
- **Cambiamento Fondamentale:** Da motore di template HTML (MVVM) a piattaforma per applicazioni web basata su **componenti** (Component Based Architecture - CBA).
- **Costruzione del DOM:** Angular *costruisce* il DOM da zero, partendo da frammenti HTML e CSS indipendenti.

Concetti Chiave di Angular

- **Moduli:**
  - NON i moduli Javascript.
  - Frammenti complessi (HTML, CSS, codice JS/TS).

    - Incapsulati, autonomi, autosufficienti.
    - Flusso di dati tra moduli controllato.
- **Direttive:**
    - Componenti speciali che modificano il comportamento del markup e del DOM.
    - Introdotte dal decoratore `@Directive`.
    - **Direttive strutturali:** Cambiano il contenuto del DOM (`*ngFor`, `*ngIf`).
    - **Direttive di attributo:** Arricchiscono il DOM (`[ngModel]`).
- **Componenti:**
    - Tipo di direttiva con template HTML e selettore (nuovo elemento di markup).
    - Introdotto dal decoratore `@Component`.
- **Routing:**
    - Passaggio tra macro-stati dell'applicazione.
    - Modifica l'URI.
    - Gestisce scoping e passaggio dati.
- **Typescript:**
    - Linguaggio adottato, compilato in Javascript a runtime.

Angular: Setup e Bootstrap

- **AngularJS:** Direttive in un documento HTML (`ng-app`, `ng-controller`).
- **Angular:** Command Line Interface (CLI) per creare progetti e componenti.
    - Struttura di directory e file predefinita.
    - Componenti in directory indipendenti (HTML, TS, CSS).

Angular.js vs. Angular: Riepilogo

- **Angular** è "opinionato": vincola il progettista al suo modello.
- **Dependency injection** e **unit testing** sono concetti fondamentali.
- Evoluzione dal templating/MVC a un'architettura a componenti.

React.js

- **Nascita e Storia:** Nasce in Facebook (2011), open source dal 2013.
- **Evoluzione:** Da motore di template a framework applicativo (browser) e ambiente di progettazione integrato (React Native, 2015).
- **React Native:** Usa Javascript, ma renderizza view native (non HTML/DOM).

Principi Base di React

- **Virtual DOM:**
    - React modifica una *copia* del DOM.
    - La libreria controlla la propagazione e aggiorna il DOM reale (reconciliation).
- **One-way data binding:**
    - Elementi del DOM associati a uno "store".
    - Modifiche allo store -> rendering del VDOM -> aggiornamento del DOM.
- **Universal rendering:**
    - Il VDOM permette l'uso di renderer diversi (non solo browser).
    - **React Native:** applicazioni native per dispositivi mobili.
    - **Next.js:** Server-Side Rendering (SSR) per SEO e performance.
- **JSX:**
    - Mix di Javascript, XML e CSS.

    - Semplifica la creazione di template HTML con espressioni Javascript.
    - **Babel:** componente che gestisce la coesistenza di JS e markup.
- **Components:**
    - Unità indipendenti di codice, markup e stile.
    - Riutilizzabili e configurabili tramite "props".

React: Modello di Processo

- Un componente React è una funzione che restituisce HTML/JSX.
- `render()` : contiene il template.
- **JSX:** mescola Javascript e markup HTML.
- Struttura di markup riutilizzabile (nuovi tag).

JSX e Babel: Dettagli

- **JSX:** estensione di Javascript con markup.
- **Babel:** interpreta ricorsivamente il codice JSX:
    - Valuta espressioni Javascript.
    - Esegue funzioni/classi per elementi corrispondenti.
    - Sostituisce l'elemento con l'output.
- `ReactDOM.render()` : inserisce il codice JSX interpretato nel DOM.

Vue.js

- **Nascita e Storia:** Nasce in Google (2014) come versione semplificata di Angular.
- **Caratteristiche:**
    - Template, sintassi handlebar, espressioni Javascript, binding bidirezionale, direttive (componenti).
    - *Non* ha: scope, dependency injection, controller.

Vue.js: Concetti Chiave

- **Interpolazione:**
    - Struttura dati unica per tutti i dati visibili.
    - Formule di interpolazione accedono direttamente a questa struttura.
- **Reattività:**
    - Interazioni utente -> funzioni di callback -> aggiornamento dati e DOM.
- **Integrabilità:** Si integra facilmente in HTML tradizionale.
- **Composition API vs Option API**:
    - Options API: Obsoleta
    - Composition API: Più moderna, consigliata

Vue.js: Single File Component

- Componenti isolati e indipendenti.
- Contengono:
    - Template HTML.
    - Stili CSS locali.
    - Dati.
    - Metodi locali.

Componenti e Binding in Vue

- Un componente può essere associato a un elemento di markup.
- `props` : per passare dati ai componenti.
- `<slot>` : per annidare contenuti.

Web Components

- **Ispirazione:** Prendono spunto dai framework a componenti (Angular, React, Vue).
- **Standard HTML:** Introducono i web component nella sintassi standard.

Caratteristiche dei Web Components

- **Custom element:** Estensione del vocabolario HTML.
- **Shadow DOM:** Mini-DOM protetto (script, stili, eventi separati).
- **HTML template:** Frammento HTML associato a un custom element.

Differenze con Altri Framework

- Registrazione del componente (array globale `customElements`).
- Shadow DOM: per elementi non (ancora) visualizzati.
- `attachShadow()` : integra lo shadow DOM nel DOM reale.
- Separazione pagina/componenti, template, shadow DOM: facoltativi.
- `<slot>` : per annidare sotto-componenti.

Altri Framework

- **Preact:** Compatibile con React, ma più piccolo e senza build.
- **Svelte:** Richiede build, supporta Javascript e Typescript, usa un linguaggio simile a JSX.
- **HTMX:** Manipolazione del markup tramite attributi (simile a Bootstrap), scambio di dati tramite HTML (non JSON).
- **Alpine:** Alternativa moderna a jQuery, minimo overhead, nessuna build.

Conclusioni

- **Component-Based Architecture (CBA):** Modello dominante.
- **HTML e CSS come "assembler":** Linguaggi di basso livello per la presentazione.
- **Ruolo delle aziende:** Influenza su WHATWG e W3C (librerie proprietarie, CDN).
- **Importanza della "conquista" degli sviluppatori:** Adozione di framework e linguaggi.