ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Ultimi concetti

Fabio Vitali + Vincenzo Rubano

Corso di laurea in Informatica
Alma Mater – Università di Bologna

Ultima lezione del corso!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Strumenti a riga di comando per lo sviluppo Web

Ambienti di sviluppo

Per ogni applicazione, distinguiamo almeno due ambienti di esecuzione:

- Ambiente di sviluppo (o locale), ovvero la macchina su cui viene implementata;
- Produzione, ovvero l'ambiente in cui l'applicazione viene eseguita quando vi accede l'utente finale.

In alcuni casi può essere presente il cosiddetto ambiente di «staging», in cui le nuove funzionalità vengono testate prima del «deploy»

CLI: di cosa si tratta?

Script, programmi e/o librerie da utilizzare mediante un terminale (appunto «a riga di comando) o mediante un IDE con uno scopo preciso, nel caso specifico per risolvere un problema particolare legato allo sviluppo Web. Pensati per migliorare la produttività del programmatore.

CLI: come si usano?

Nella maggior parte dei casi, vengono eseguiti nell'ambiente di sviluppo sul codice sorgente dell'applicazione; nell'ambiente di produzione, invece, viene caricato il codice risultante dall'esecuzione di questi strumenti.

Questa è una regola generale, ma ovviamente vi sono eccezioni!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Sopravvivere all'evoluzione

Lo sviluppo front-end si evolve

Al contrario di quanto si possa pensare, i linguaggi che utilizziamo quotidianamente per sviluppare applicazioni Web (soprattutto JavaScript e CSS, ma anche HTML) sono in continua evoluzione secondo particolari meccanismi ben definiti. Le novità introdotte sono spesso «desiderabili» durante lo sviluppo, ma purtroppo non è detto che esse siano supportate negli ambienti di esecuzione (i browser, Node.JS, etc) in cui l'applicazione debba essere eseguita. Ciò può avvenire sia perché le novità non sono ancora state implementate, sia perché è necessario supportare versioni precedenti degli ambienti di esecuzione rispetto a quelle che ne integrano il supporto; come fare, allora?

CanIUse

Il sito CanIUse (www.caniuse.com) è una fonte di informazioni preziosissima! Esso, infatti, indica a partire da quale versione di ogni browser una certa funzionalità (sia essa di HTML, JavaScript o CSS) è supportata, nonché una stima del «market share» di ciascun browser. I dati sono messi a disposizione anche in modo strutturato, tramite delle comode API!

CSS DB

- Contiene una lista delle feature relative ai fogli di stile CSS, ed il loro grado di «maturità» nel processo di integrazione nello standard vero e proprio (che è diviso in più fasi).

Browserslist

Descritto come «The config to share target browsers and Node.js versions between different front-end tools", si tratta di uno strumento "molto potente" per descrivere il target (browser e version di node) che si intendono supportare.

Queste informazioni vengono utilizzate da altri strumenti per determinare se una certa "caratteristica" del codice sorgente è supportata da tutti i browser (sfruttando caniuse e cssdb), determinare il grado di supporto (complete, parziale, con bug più o meno seri, etc), e decidere se e come trasformare il codice, o usare dei polyfill.

I polyfill

In programmazione Web, per «polyfill» si intende un frammento di codice che implementa funzionalità che non fanno parte di quelle offerte da un browser, aggiungendo caratteristiche che ci si aspetta siano disponibili ma non lo sono.

La necessità di un polyfill può essere dovuta a vari fattori, tra cui la assenza di un'implementazione originale, il fatto che essa contenga bug critici o che sia incompleta.

Molti polyfill utilizzati in JavaScript sono forniti dalla libreria "core-js".

Problemi risolti

Esistono strumenti per risolvere i problemi più disparati, ma oggi ci occuperemo di:

- gestione delle dipendenze;
- CSS;
- Type checking;
- Linting;
- Transpilation;
- Minification;
- Bundling

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Gestione delle dipendenze

Cos’è la gestione delle dipendenze

Nel mondo della gestione del software esiste un luogo terrificante chiamato “dependency hell (inferno delle dipendenze)”. Quanto più cresce il vostro sistema e quanti più pacchetti integrate nel vostro software, tanto più facilmente vi ritroverete, un giorno, in questa valle di lacrime.

[Cit.: Versionamento Semantico 2.0.0 | Semantic Versioning (semver.org)]

Strumenti per la gestione delle dipendenze

Si tratta di strumenti che facilitano la gestione di tutti i pacchetti necessari allo sviluppo e/o all'esecuzione di un'applicazione Web. Il loro funzionamento è solitamente molto simile: ci limiteremo ad esaminare npm e yarn.

Automatizzano la gestione di molti aspetti problematici, come la selezione di versioni compatibili delle dipendenze.

DevDependencies VS. dependencies

- DevDependencies: sono necessarie per lo sviluppo dell'applicazione, ma non per la sua esecuzione nell'ambiente di produzione. Esempio: vue-cli.
- Dependencies: sono indispensabili per l'esecuzione dell'applicazione in ambiente di produzione, e ne consentono il funzionamento. Esempio: vue.

NB: quando si aggiunge una dipendenza, di fatto si aggiungono al Progetto anche tutte le sue dipendenze (chiamate "transitive-dependencies"). Tutte le dipendenze di un Progetto formano il cosiddetto "grafo delle dipendenze".

NPM e Yarn

Sono due gestori di pacchetti molto simili pensati per gestire le dipendenze in ambiente node.js, ma consentono anche la creazione di comandi personalizzati per l'automazione di task ripetitivi.

Si basano su due file:

- package.json, che contiene i metadati del progetto, tutte le dipendenze (dev e non) ed eventualmente una specifica delle versioni accettabili;
- Un file «lock» (package-lock.json e yarn.lock, rispettivamente), che contiene l'elenco delle dipendenze e per ciascuna di esse quale versione è stata installata in un certo istante.

Perché il lock?

Un file «lock» è necessario per rendere l'installazione «riproducibile», ovvero consentire ad un altro utente di installare le stesse identiche dipendenze con cui è stata testata l'applicazione su un'altra macchina. Le versioni accettabili, infatti, possono essere specificate sotto forma di range per agevolare gli aggiornamenti e l'installazione di versioni «compatibili».

Alcuni comandi

Inizializzare un progetto:

- `npm init`
- `yarn init`

Aggiungere una dipendenza:

- `npm install react`
- `yarn add react`

Aggiungere una dipendenza dev:

- `npm install react -include=dev`
- `yarn add --dev react`

Installare tutte le dipendenze di un progetto:

- `npm install`
- `yarn install`

Aggiornare tutte le dipendenze del progetto:

- `npm update`
- `yarn upgrade`

Il problema delle dipendenze

Dato un progetto con n dipendenze, vogliamo aggiungere una nuova dipendenza in modo tale che:

1. essa (e tutte le sue eventuali dipendenze) siano compatibili con le dipendenze già necessarie;
2. Sia possibile aggiornare tutte le dipendenze alle ultime versioni disponibili, garantendo però la loro compatibilità.

In ambito node.js, sfruttiamo il semantic versioning e la specifica dei «range accettabili», ovvero indicare quali versioni di una dipendenza sono accettabili nel progetto all'istante attuale e «nel futuro».

Semantic versioning

Quando si pubblica un pacchetto, di solito si specificano le versioni nel formato x.y.z dove:

- Un incremento nel valore di x (detto «major») indica che la versione introduce cambiamenti nelle API pubbliche non retrocompatibili con le versioni precedenti;
- Un incremento nel valore di y (detto “minor”) indica che la nuova versione introduce cambiamenti nelle API retrocompatibili con le version precedenti;
- Un cambiamento nel valore di z (detto “patch”) indica che la nuova versione si limita a risolvere bug.

Esempi di range

In package.json:

"mylib": "1.0.0"

Indica che può essere accettata solo la versione 1.0.0

"mylib": "1.0.x"

Indica che possono essere accettate tutte le versioni «patch» (i.e. >=1.0||<1.1).

"mylib": "1.x"

Indica che possono essere accettate tutte le versioni patch e minor (i.e. >=1.0||<=2.0).

NB: || rappresenta l'operatore or, e può essere usato per esprimere range.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CSS

CSS Preprocessors

Strumenti che consentono di convertire codice scritto in un apposito linguaggio di scripting in fogli di stile CSS veri e propri. Creare un linguaggio di scripting consente di «estendere» le funzionalità rese disponibili da CSS mettendo a disposizione concetti quali gli operatori logici, le variabili, i mixins, l'ereditarietà, le funzioni e gli operatori matematici.

I più comuni sono: SASS, LESS e stylus.

SASS

SASS (Syntactically Awesome Style Sheet) è il primo CSS preprocessor, è stato rilasciato nel 2006; inizialmente richiedeva l'utilizzo di Ruby, ma nel tempo è stata implementata la libreria «libsas» che, essendo scritta in C, consente di interpretare questo linguaggio praticamente in ogni ambiente (node.js, php, etc).

I file hanno due possibili estensioni: .sas, che utilizza la sintassi originale (basata su indentazione), e «.scss», che utilizzano una sintassi CSS-like ma arricchita: ogni file .css è automaticamente un file .scss, ma non un file .sas.

LESS

LESS (Leaner Style Sheets) è stato rilasciato nel 2009. Implementa molti dei concetti presenti all'interno di SASS, e nel corso del tempo i due tool si sono influenzati a vicenda (la seconda sintassi di SASS è molto simile alla sintassi di LESS, per esempio).

Viene fornito sotto forma di libreria JavaScript eseguibile in ambiente Node, ma è disponibile anche una versione eseguibile direttamente nel browser.

PostCSS

Descritto sul sito ufficiale come «A tool for transforming CSS with JavaScript", PostCSS permette di trasformare I fogli di stile in ambiente Node eseguendo diverse operazioni:

- Introdurre automaticamente i "prefissi" laddove necessari (*autoprefixer*), per abilitare attributi sperimentali;
- Utilizzare sintassi CSS non ancora pienamente supportata dai browser (*postcss-preset-env*), trasformandola in costrutti equivalenti e supportati;
- Eseguire automaticamente il "linting" dei file CSS (*stylelint*), per verificare il rispetto di regole stabilite (dallo sviluppatore del progetto che lo usa);
- Usare i cosiddetti "*CSS modules*", per avere uno "scoping locale" per i selettori (e non globale, come avviene di default);
- ... e molto altro: può essere esteso tramite plugins, e ne esistono di ogni tipo: conversion di unità, embedding di grafica, rimozione di CSS non necessario dato un certo markup, etc.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Linting

Cos’è il linting?

Con «linting» si intende il processo di analizzare staticamente il codice sorgente scritto in un certo linguaggio alla ricerca di potenziali «errori». In particolare, nella programmazione web ciò spesso si traduce in un’analisi per individuare violazioni di regole stabilite dallo stesso programmatore del progetto.

Tali regole possono riguardare aspetti stilistici (indentazione, ritorni a capo, etc.), così come l’uso (o il divieto di usare) certi costrutti.

eslint

Eslint (the pluggable JavaScript linter) è uno dei linter più usati in ambiente Web. Il suo punto di forza sta nell'avere un «core» minimale, e poter essere esteso mediante plugin.

Il core ed i vari plugin forniscono un elevatissimo numero di regole, lasciando al programmatore la facoltà di decidere (attraverso un file di configurazione) quali utilizzare e quali no.

Siccome il numero di regole è molto elevato (nell'ordine di centinaia), esistono dei «preset» per renderne più facile l'applicazione.

Eslint – categorie di regole

In eslint, le regole sono raggruppate per tipologia:

- Possibili errori, ovvero regole che evitano di commettere errori logici e/o di sintassi;
- Best practice, relative ai migliori modi per implementare determinate funzionalità in modo da evitare problemi;
- Regole relative alle dichiarazioni di variabili;
- Regole stilistiche, relative allo stile di programmazione ed alla formattazione del codice; possono essere soggettive.
- Es6, regole relative all'uso di funzionalità introdotte in EcmaScript 6.

eslint --fix

Un altro punto di forza di eslint è quello di poter risolvere «automaticamente» molti dei problemi individuati, grazie al meccanismo denominato «autofix».

Tale meccanismo deve essere abilitato esplicitamente (nel file di configurazione o a riga di comando), giacchè potrebbe creare problemi con alcuni editor.

Prettier

Prettier («the opinonated code formatter») è uno strumento specializzato nell'eseguire il parsing del codice sorgente, e stamparlo applicando regole di formattazione ben precise.

Supporta molteplici sintassi (typescript, JavaScript, JSX, JSON).

Al contrario di quanto avviene con eslint, le sue regole si basano su «opinioni» ben precise sul modo con cui dovrebbe essere scritto il codice, ed offre meno flessibilità a livello di configurazione.

Un esempio

```
foo(reallyLongArg(), omgSoManyParameters(),
IShouldRefactorThis(), isThereSeriouslyAnotherOne());
```

Sebbene questo codice sia perfettamente valido sintatticamente, la sua leggibilità è quantomeno discutibile… Può essere riformattato come:

```
foo(
  reallyLongArg(),
  omgSoManyParameters(),
  IShouldRefactorThis(),
  isThereSeriouslyAnotherOne()
);
```

Ben più leggibile!

Perché tanta importanza allo stile?

Avere uno stile di programmazione «coerente» rende più agevole la lettura del codice. Ogni programmatore, però, tende ad applicare le regole che più preferisce, sempre che si preoccupi di questi aspetti. Ecco perché questi tool sono importanti!

Essi possono essere integrati in vari IDE, utilizzati a riga di comando, o addirittura diventare parte delle pipeline della cosiddetta «continuous integration (CI)», molto comune nei progetti open source.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Type checking

Esempio I

```
const el = document.getElementById('my-id')
el.classList.add('foo')
```

Il codice non è sempre corretto: se l'elemento con my-id non esiste, infatti, viene sollevata un'eccezione e l'esecuzione viene interrotta: qualunque istruzione successiva nello stesso blocco non sarà valutata, e senza una corretta gestione delle eccezioni potrebbe essere sospesa l'esecuzione dell'intero script.

Esempio II

```
function add(a, b) {
  return a+b
}

add(3, 4)
Add(3, ‘4’)
```

Cosa viene restituito?

I tipi in JavaScript

Per sua natura, il linguaggio JavaScript presenta un sistema di tipi dinamico, in cui cioè la tipizzazione viene eseguita durante l'esecuzione del codice.

Questo rende difficile il debugging, giacchè non è possibile eseguire un'analisi statica dei tipi… o per lo meno essa è resa talmente problematica da non essere praticabile in tempi brevi. Vi sono design pattern, inoltre, in cui essa è di fatto impossibile.

Se su progetti di piccola scala questo aspetto potrebbe non essere problematico, con l'aumento delle righe di codice il problema diventa sempre più pressante; è molto facile, inoltre, far sì che il codice manifesti comportamenti diversi a causa dei parametri utilizzati per invocare le funzioni.

Tutto questo, però, può essere valutato solo a runtime, eseguendo il codice in ogni possibile scenario, seguendo tutte le sue possibili ramificazioni… E presto diventa non sostenibile!

Typescript

Si tratta di un linguaggio open source sviluppato da Microsoft, il cui scopo è quello di rendere più agevole l'analisi statica dei tipi in JavaScript.

Il linguaggio è definito come un super-set di JavaScript, perciò ogni costrutto valido in JavaScript è valido in typescript; vengono però aggiunti costrutti per l'annotazione dei tipi, nonché un sistema di type-inference che consente di conoscere (staticamente) il tipo di una variabile.

Un compilatore si occupa di «tradurre» i sorgenti typescript in JavaScript vero e proprio, eseguendo nel contempo l'analisi statica dei tipi: ciò consente di evitare molti bug.

Esempio I – in typescript

```
const el = document.getElementById('my-id')
el.classList.add('foo')
```

// Errore: el è di tipo HTMLElement|NULL, ed il tipo «HTMLElement|NULL» non ha la proprietà «classList».

Soluzione: usare il «null coalescing operator», oppure controllare esplicitamente il valore restituito dalla funzione.

Esempio II – in typescript

```
function add(a: int, b: int): int {
  return a+b
}

add(3, 4) // corretto
add(3, ‘4’) // errore: il secondo parametro è una
stringa, non un intero
```

Si può anche tipizzare la funzione in modo da mantenere la «dinamicità» sfruttando la composizione dei tipi…

Tipi in typescript I

Typescript fornisce alcuni tipi primitivi, ma è possibile derivare nuovi tipi per composizione.

Consente inoltre di definire i tipi corretti per array, dizionari, insiemi e tutte le altre «collections», i «literal objects», le classi, e permette di definire interfacce e dichiararne la conformità da parte di classi e oggetti… E molto altro!

Tipi in typescript II

Le definizioni per molti tipi (API relative al DOM, per esempio) sono «integrate» nel compilatore; è inoltre possibile utilizzare i tipi anche per le librerie di terze parti.

NB: a volte è necessario installare i tipi di una libreria separatamente; ciò è dovuto al modo con cui essa è implementata. Se una libreria non fornisce nativamente i suoi tipi, è possibile che essi siano resi disponibili da sviluppatori di terze parti.

Se una libreria non contiene le sue definizioni di tipo, è molto probabile che esse siano disponibili in package il cui nome è della forma «@types/libraryname».

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Transpilation

Cos’è la transpilation?

Per transpilation si intende quel processo di conversione attraverso un programma (detto transpiler) di un codice sorgente scritto in un certo linguaggio in una sua forma equivalente scritta in un altro linguaggio.

A differenza di un compilatore, il transpiler opera con due linguaggi il cui livello di attrazione è molto simile.

Babel I

Si tratta di un transpiler (a volte detto anche compilatore) per il linguaggio JavaScript, il cui scopo primario è quello di consentire l'utilizzo delle funzionalità più moderne introdotte nel linguaggio indipendentemente dal supporto offerto tramite i browser.

Ciò viene ottenuto con diverse tecniche:

- Trasformando la sintassi dei costrutti;
- Iniettando i polyfill necessari.

Babel II

Esso è uno strumento molto potente, ma per sfruttarlo al massimo richiede una configurazione opportuna; può svolgere al meglio il suo compito in combinazione con altri tool (core-js, browserlist, etc).

Esistono plugin che offrono il supporto per i principali framework: react (responsabile della traduzione della sintassi JSX in JavaScript, per esempio), Vue (per trasformare i single file template ed altri costrutti), etc.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Minification

Cos’è la minification?

In programmazione, per «minification» si intende il processo di rimozione di tutti i caratteri non necessari dal codice sorgente di uno script scritto in un linguaggio interpretato, o da un contenuto scritto in un linguaggio di markup, senza alterarne le funzionalità (o la semantica).

Lo scopo è quello di «minimizzare» la dimensione del codice sorgente, così da velocizzarne la trasmissione tramite una rete (come per esempio Internet); di solito si rimuovono i caratteri non necessari spazi bianchi, ritorni a capo, etc), ma possono essere anche eseguite «trasformazioni» che convertano determinati costrutti in forme equivalenti «più brevi».

Minification vs. compressione

La minification è diversa dalla compressione, in quanto il codice sorgente «minificato» può essere interpretato nal pari della sua forma «non minimizzata», senza cioè richiedere step aggiuntivi (come una fase di decompressione).

minifier

Esistono diversi strumenti più o meno specializzati nella minification di sorgenti scritti in vari linguaggi (HTML, JavaScript, CSS). Vediamo all'opera terser, uno dei minifier più utilizzati per ridurre le dimensioni del codice JavaScript.

Esempio di minification I

Consideriamo questo semplice frammento di codice JavaScript:

```
/**
 * Adds the given class to each DOM node in the given list
 *
 * @param elements: DOM nodes to add the class to
 * @param classname: CSS class to be added
 */
function addClass(elements, classname) {
  elements.forEach((element) => {
    element.classList.add(classname)
  })
}

const paragraphClass = 'para'
const paragraphs = document.querySelectorAll('p')
addClass(paragraphs, paragraphClass)
```

Esempio di minification II

Minificato mediante terser può essere convertito semplicemente in:

```
document.querySelectorAll("p").forEach((a=>{a.cla
ssList.add("para")}));
```

Le due forme in cui è scritto il codice (non-minificata e minificata), sono del tutto equivalenti; mentre la prima occupa 415 bytes, la seconda ne occupa “solo” 71, che è inferiore dell’83%!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Bundling

Cos’è il bundling?

Quando si sviluppa un’applicazione Web, per favorire la manutenibilità del codice è prassi comune «suddividere» il codice in più file.

Quando l’applicazione deve essere eseguita, però, avere molti file di piccole dimensioni può avere un impatto negativo sulle performance: il *bundling*, dunque, è quel processo che consiste nell’unificare più file dello stesso tipo (CSS, JS, etc.) in un unico file secondo criteri.

Tale «raggruppamento» può avvenire in base a diversi criteri.

Il bundling non è un problema banale

Dal punto di vista implementativo, il bundling non è un problema affatto banale: specialmente quando si parla di JavaScript, infatti, occorre considerare diversi aspetti quali l'ordine di esecuzione (ed i side-effects), i potenziali conflitti di nomi (classi, variabili, metodi, etc), le dimensioni dell'output.

Esistono ovviamente diverse tecniche per affrontare questi problemi…

I bundler

Esistono molteplici bundler in circolazione.

Sebbene molti di essi condividano un certo insieme di funzionalità, ognuno può presentare peculiarità più o meno significative…

Ci limiteremo ad esaminarne un paio: webpack e rollup.

webpack

Bundler molto potente e flessibile, la cui configurazione può essere semplici o (molto) complessa a seconda dei casi d'uso. In generale, crea un «grafo delle dipendenze» del sorgente del progetto e, dopo aver eventualmente trasformato il codice (aggiornando il grafo opportunamente), procede alla creazione di uno o più artefatti secondo parametri configurabili.

rollup

Nel corso del tempo, rollup si è fatto conoscere come un'alternativa a webpack più efficiente, più semplice da configurare, più performante (il bundling richiede tempo!), e per la generazione di artefatti più efficienti e di dimensioni inferiori, grazie all'uso di tecniche innovative (e.g. scope hoisting).

Sebbene anche l'ecosistema intorno a rollup si stia evolvendo velocemente, spesso non è tanto maturo quanto webpack, perciò ci si potrebbe imbattere in vari bug da "aggirare" in modi non sempre banali, in attesa che vengano risolti.

Al contrario di webpack, inoltre, il suo core è estremamente minimale: ciò è sia un vantaggio che uno svantaggio.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Putting it all together…

Una pipeline

Gli strumenti a riga di comando disponibili sono molteplici, e spesso complementari. Realisticamente, a parte doverose eccezioni, è difficile che essi vengano utilizzati «da soli»; più probabile, invece, è che vengano organizzati in una «build pipeline», in modo tale che il codice sorgente venga controllato, «compilato» e «assemblato» secondo una certa configurazione.

Esempio di pipeline

Esaminiamo una possibile pipeline per file JavaScript. Ogni passaggio può generare sia errori che warning, i quali possono interrompere il processo.

- Partiamo dal file principale;
- Analizziamo le sue dipendenze (ricorsivamente);
- Processiamo i file, eseguendo linting e/o type-checking;
- Eseguiamo il transpiling di ogni codice (traducendo costrutti, iniettando polyfill in base ai target supportati, etc);
- Creiamo il «bundle» risultante, da eseguire sul browser.

Configurazione della pipeline

Queste pipeline possono diventare piuttosto complesse; di solito, esse vengono gestite attraverso un insieme di configurazioni per ottenere un risultato organico.

Spesso, inoltre, vi è un «overlapping» tra vari tool che offrono la stessa funzionalità (esliint e prettier, eslint e typescript, per esempio); è compito dello sviluppatore trovare la configurazione ideale, e non è sempre banale.

Perciò…

Task management

Le tipologie di strumenti che abbiamo analizzato sono solo una parte di quelle disponibili, ma già fanno emerge l'esigenza di eseguire (e coordinare l'esecuzione) di vari compiti: come fare?

Come detto, spesso si lascia al bundler il compito di coordinare vari aspetti; esistono però altre possibilità, e non sono mutualmente esclusive:

- Usare i cosiddetti «npm scripts», che consentono di eseguire comandi complessi definiti all'interno del file «package.json» con alias più semplici;
- Usare i cosiiddetti task-runner, come grunt e gulp.

*-cli

Quello che spesso accade, dunque, è che i principali framework più usati per lo sviluppo Web mettano anche a disposizione delle pipeline già preconfigurate, offrendo un processo «ottimizzato» per quel determinato framework; è il caso di angular-cli, react-cli, vue-cli, per esempio.

Sarebbe importante capire il funzionamento di questi strumenti, però, così da evitare di vedere i command line tools come una «blackbox»: di fatto tutto è configurabile, e non è detto che le configurazioni fornite dai framework soddisfino sempre le nostre esigenze o siano prive di bug.

E' quindi importante sapere come/dove mettere le mani per risolvere!

Le source maps

Quando il codice viene trasformato, fare il debugging diventa complicato: come si determina a quale frammento del sorgente originale corrisponde il frammento di codice in esecuzione?

Si usano le cosiddette «source map», file che consentono di mappare frammenti del codice trasformato (detto «generated code») ai frammenti del codice sorgente (detto «original code») che li hanno generati.

Nel file generato, di solito, si inserisce l'URL della source map.

Related resources I

- npm (npmjs.com)
- GitHub - yarnpkg/yarn: The 1.x line is frozen - features and bugfixes now happen on https://github.com/yarnpkg/berry
- Semantic Versioning 2.0.0 | Semantic Versioning (semver.org)
- Sass: Sass Basics (sass-lang.com)
- Getting started | Less.js (lesscss.org)

Related resources II

- PostCSS - a tool for transforming CSS with JavaScript
- GitHub - csstools/postcss-preset-env: Convert modern CSS into something browsers understand
- GitHub - css-modules/css-modules: Documentation about css-modules
- Cssdb
- Can I use... Support tables for HTML5, CSS3, etc

Related resources III

- ESLint - Pluggable JavaScript linter
- TypeScript: Typed JavaScript at Any Scale. (typescriptlang.org)
- Webpack
- rollup.js (rollupjs.org)
- GitHub - terser/terser:  JavaScript parser, mangler and compressor toolkit for ES6+

Related resources IV

- Grunt: The JavaScript Task Runner (gruntjs.com)
- gulp.js (gulpjs.com)
- Vue CLI (vuejs.org)
- Creare una Nuova App React – React (reactjs.org)
- Angular - CLI Overview and Command Reference

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Node e gocker al DISI

Usare node sulle macchine del DISI

- Il progetto di TW richiede l'uso di node come motore dell'applicazione server-side.
- Inoltre le regole del corso impongono l'uso di una macchina DISI per ogni servizio server-side del progetto.
  - Ogni struttura dati e ogni servizio sw creato da studenti UniBo deve essere fornito da una macchina *.cs.unibo.it
- Il DISI mette a disposizione DUE modi per attivare un servizio node:
  - attraverso *gocker*, un servizio docker di virtualizzazione di una macchina online (*container*)
  - Via linea di comando su una porta alta (>1024)
- Entrambi richiedono qualche competenza di linea di comando (poca roba)
- Entrambi funzionano sullo spazio dati e con i permessi dello studente, e non dell'amministratore o di root.

Studenti NON di Informatica / Informatica per il Management

Per essere ammessi ad usare le risorse digitali del dipartimento (tra cui, appunto, gocker):

- Gli studenti di Informatica e Informatica per il Management non debbono fare nulla
- Gli studenti di altri dipartimenti (Lettere, Filosofia, Comunicazione, Storia, Giurisprudenza, ecc.) debbono mandare una mail a tecnici@cs.unibo.it (e in cc a me) dal loro account istituzionale in cui dichiarano di star seguendo il mio corso e di voler avere accesso alle risorse informatiche del dipartimento DISI per la durata dell'esame.
- Meglio se raccogliete tutte le richieste in un'unica mail cumulativa.
- Nel giro di pochi giorni riceverete una comunicazione che vi è stato dato accesso e potete richiedere le risorse del dipartimento

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Uso della shell Unix per chi non ricordasse Sistemi Operativi

Un minimo di sintassi shell (1/3)

Alla shell delle macchine si accede via ssh:

```
> ssh eva.cs.unibo.it
fabio@eva.cs.unibo.it's password:
```

O equivalentemente:

```
> ssh fabio.vitali@eva.cs.unibo.it
fabio.vitali@eva.cs.unibo.it's password:
```

Appare il prompt della shell, che varia da caso a caso:

```
nomeutente@nomemacchina:dir$
fabio@eva:/home/web/siteXXXX$
```

Un minimo di sintassi shell (2/3)

- Per cambiare directory:
  - cd /path/assoluto `fabio@eva:~$ cd /home/web/siteXXXX`
  - cd nomelocale `fabio@eva:/home/web/siteXXXX$ cd html`
- Per mostrare il contenuto della directory:
  - ls (lista semplice dei file visibili della directory corrente)
  - ls -l (lista completa dei file visibili della directory corrente)
  - ls –al (lista completa dei file visibili e invisibili della directory corrente)
  - `fabio@eva:/home/web/siteXXXX$ ls -al`

```
total 308
drwxr-xr-x    7 root     root       4096 May  4 09:26 .
drwxr-xr-x 3082 root     root     286720 May  4 15:48 ..
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 cgi-bin
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 data
drwxrwsr-x    3 site1833 site1833   4096 May  8 16:31 html
drwxr-x---    2 webadm   site1833   4096 May  4 09:26 log
drwxrwsr-x    2 site1833 site1833   4096 May  4 09:26 scripts
```

Un minimo di sintassi shell (3/3)

- Per cambiare i permessi di accesso:
  - chmod permessotesto nomefile `chmod a+x file.php`
  - chmod permessobinario nomefile `chmod 755 file.php`
- Per modificare un file di testo:
  - nano nomefile o pico nomefile
  - vi nomefile oppure vim nomefile
  - emacs nomefile
  - Sono tutti editor molto diversi tra di loro. *Io uso vi* .
- Per avere istruzioni su un comando Linux
  - man comando `man ls`
  - man –k keyword `man -k files`

Imparate la shell!
Imparate ad usare i vostri strumenti

Macchine GNU/Linux

- Le macchine hanno nomi di personaggi di opere.
- Alcuni esempi:
  - annina.cs.unibo.it
  - berta
  - ceprano
  - dorina
  - eufemia
  - frazier
  - giovanna
  - …
- L'elenco completo è su: http://www.informatica.unibo.it/it/servizi-informatici/accesso-da-remoto

Il file system condiviso del DISI

Tutte le macchine Linux del dipartimento hanno una struttura del file system simile, e condividono alcune directory:

- /home
- /public

Questo significa che da qualunque macchina vi colleghiate, potrete sempre accedere alla stessa directory.

Dentro ad /home si trovano le home directory di utenti e progetti e siti web, come segue:

- /home/faculty, /home/guest-fac, /home/phd-students, /home/postdoc, /home/staff, /home/students: home directory di specifici utenti umani, a seconda del ruolo e della durata del diritto d'accesso.
- /home/esami, /home/projects: home directory di progetti di ricerca e strutture temporanee di vario tipo, attivate e disattivate secondo necessità
- /home/nws, /home/web: home directory di siti web, le uniche su cui sia attivato un server http

La directory /home/web/site2021XX/

- Contiene cinque directory:
  - cgi-bin
  - data
  - html
  - log
  - scripts
- Di queste, ci serve SOLO html, che è il posto dove mettere script e file del nostro sito web.
- Ad /home/web/site2021XX/html/ corrisponde la directory di base di http://site2021XX.tw.cs.unibo.it/ .

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Node e npm su Unix

Docker

Node da linea di comando

```
var http = require('http');

http.createServer(
   function (request, response) { ... }
).listen(8000);
```

- Il modo più semplice per attivare node è da linea di comando:
  - `fabio@eva:~$ cd /home/web/siteXXXX/html/`
  - `fabio@eva:/home/web/siteXXXX/html/$ node index.js`
- A questo punto node risponde all'indirizzo:
  - `http://eva.cs.unibo.it:8000/`

PATH e shell

- Può succedere che otteniate errori tipo

```
fabio.vitali@eva:~$ node index.js
-bash: node: command not found
fabio.vitali@eva:~$ npm install express
-bash: npm: command not found
```

- Questo significa che non avete la directory di node e npm nel PATH (che sono le directory cercate automaticamente dalla shell).
  - E' necessario o specificare il path assoluto di node e npm...
    - `/usr/local/node/bin/node index.js`
    - `/usr/local/node/bin/npm install express`
  - oppure inserire la directory di node nel PATH:
    - `cd $home`
    - `nano .bash_profile`
    - Aggiungete la seguente riga: export PATH=/usr/local/node/bin/:$PATH
    - Riavviate la shell, lanciando il comando: exit

Limiti e problemi di node da linea di comando

- Funziona solo finché la shell è aperta.
  - Quindi non possiamo scollegarci finché vogliamo che funzioni
  - Non è vero, c'è un modo, ma non è il momento per parlarne.
- Funziona solo dalla macchina da cui abbiamo lanciato l'eseguibile.
  - Quindi l'URI deve specificare esattamente la macchina scelta.
- Funziona solo su porte alte > 1024, e la porta deve essere libera per quella macchina.
  - quindi l'URL da usare DEVE specificare la porta
  - Suggerimento: poiché Gocker funziona necessariamente sulla porta 8000, usate la porta 8000.
- Ci può essere solo UN servizio attivo su una data porta e un dato server
  - Quindi se usate tutti la porta 8000 dovete tutti usare server diversi.
- Utile nelle fasi di programmazione e debug.
- Una volta che il server funzioni dovrete cercare qualcosa di più stabile.

Docker (1/3)

- E' un fornitore di container software, ovvero macchine virtuali minimali in grado di eseguire una sola applicazione senza richiedere tutta l'infrastruttura di un sistema operativo.
- Il DISI utilizza docker per fornire macchine virtuali con un software preinstallato:
  - node
  - python
  - php
  - mongoDB
- **Sebbene in teoria sia possibile, questo laboratorio non accetta immagini con sw preinstallato diverso da questi.**
- I container docker accedono al file system condiviso dei server del dipartimento, quindi possono essere attivati su qualunque directory del file system. Noi tradizionalmente usiamo /home/web/site2021XX/html/

Docker (2/3)

- La nostra implementazione di docker risponde via ssh all'indirizzo gocker.cs.unibo.it. E' possibile accedere via ssh a gocker solo da una macchina di laboratorio. Si deve dunque accedere via ssh prima ad una macchina di laboratorio (es. eva.cs.unibo.it) e da quella macchina accedere sia ssh a gocker.cs.unibo.it.
  - I siti web studenti per ragioni storiche utilizzavano una specifica macchina virtuale chiamata Golem. ***Gocker = Golem + Docker***.
  - Oggi Golem non esiste più o esiste per altri scopi, ma il nome è rimasto.
- Alla partenza Gocker fornisce una shell MOLTO minimale, senza navigazione sul file system, con pochi comandi:
  - create: crea un container con un solo servizio (node o php o python)
  - list: fornisce un elenco di tutte le directory dell'utente su cui è possibile o è stato attivato un container
  - remove: rimuovi un container
  - restart: ferma e riavvia un container
  - exit: esci dalla shell di docker
  - help: mostra un semplice aiuto sui comandi

Ottenere un docker

- Collegarsi su https://ssl.cs.unibo.it/csservices/
- Chiedere un nuovo servizio, specificando tutti i membri del team come corresponsabili, e me come docente
- Attendere una mail di conferma dell'attivazione del servizio.

La shell di docker

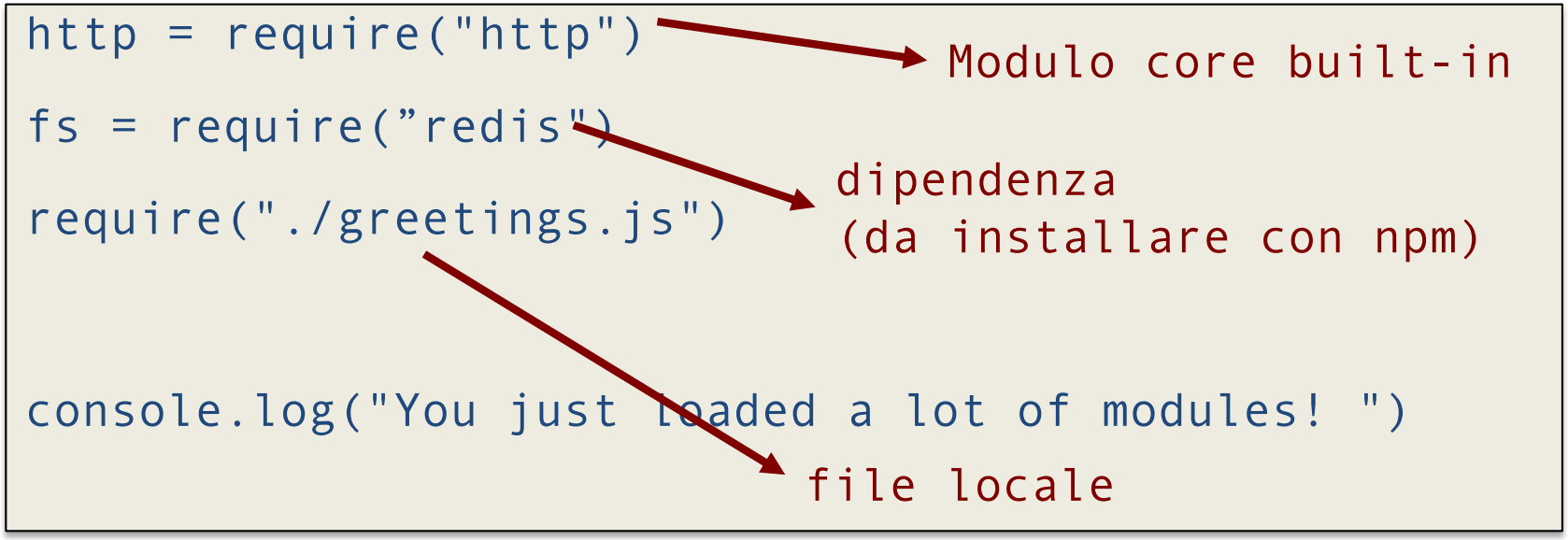
Il comando più interessante è create:

- `create <image> <site> [<script>]`
- ad es.: `create node-15 site2021XX index.js`

- image è uno di vari come:
  - `static (no server-side scripting)`
  - `node-15 (raccomandato)`
  - `node-14`
  - `nodemon-15 (raccomandato)`
  - `mongo (mongodb 4.4.5, raccomandato)`
- site è il dominio di V livello associato al servizio
  - `http://site2021XX.tw.cs.unibo.it/`
- Ricordarsi che la nostra configurazione di Docker prevede che la porta aperta per node.js sia la porta 8000!
- script è utile solo per node, ed è lo script javascript staticamente associato all'applicativo node, che fornisce le funzionalità attive.

I moduli di node

- Node è estremamente modulare. E' possibile mettere codice in file javascript ed eseguirlo secondo necessità.
- Il meccanismo di caricamento di moduli di node è semplice:
  - un modulo è un file Javascript.
  - Quando si richiede un modulo, esso viene cercato localmente o globalmente secondo un modello preciso.
- I moduli vengono caricati con `require()`.



npm

- I moduli di node vengono distribuiti ed installati con npm (*node package manager*)
- npm viene eseguito via command-line e interagisce con il registro npm.
  - meccanismo robusto per gestire dipendenze e versioni dei pacchetti (moduli)
  - semplice processo di pubblicazione di pacchetti e condividerli con altri utenti.
- In più: una piattaforma per repository pubbblici e privati, servizi enterprise (integrati con firewall aziendali), integrazione con altre piattaforme, ecc.

Usare npm

Un pacchetto npm è semplicemente un insieme di file, il più importante dei quali è *package.json*, che contiene metadati sul pacchetto (contenuto, dipendenze, ecc.)

Comandi utili di npm:

- npm init
  - Genera un file package.json globale nella directory in cui si trova. Tipicamente è la directory in cui eseguire node.
  - Viene anche creata la directory node_modules in cui verranno posizionati i package installati.
- npm install [package]
  - installa il pacchetto specificato, comprese tutte le dipendenze specificate, nella directory node_modules
- npm uninstall [package]
  - rimuove il pacchetto specificato, comprese tutte le dipendenze specificate e non usate da altri pacchetti installati
- npm update [package]
  - aggiorna esplicitamente il pacchetto specificato, comprese tutte le dipendenze specificate.

Express.js

- Express è un modello di server web per node.js
  - Open-source, licenza MIT
  - molto usato e molto supportato dalla comunità
  - molto semplice e molto espandibile con plugin
- Express si dedica al routing, alla gestione delle sessioni, all'autenticazione degli utenti, e molti altri principi fondanti delle connessioni HTTP.
- Nato nel 2010, acquistato nel 2015 da IBM, poi regalato da IBM alla Node.js Foundation.

Express/cors

Permette di definire server Express che realizzano completamente le funzionalità CORS (*Cross Origin Resource Sharing*).

- Per ragioni di sicurezza, un documento HTML può ricevere risorse (immagini, dati Ajax, script, ecc.) solo da un server posto nello stesso dominio del documento HTML di origine (*Same Origin Policy*).
- E' a discrezione del server permettere ad un'applicazione di un dominio diverso di richiedere le proprie risorse (*Cross Origin Resource Sharing*).
- Il browser, subito prima di eseguire un collegamento Ajax ad una risorsa di un dominio diverso, esegue un OPTION al server.
- Affinché sia poi possibile eseguire la richiesta, il server deve rispondere con alcune intestazioni HTTP specifiche, altrimenti il browser non effettuerà la richiesta vera e propria.
- Tipicamente: `Access-Control-Allow-Origin: *`

Il package express/cors inserisce automaticamente la risposta cors più appropriata e gestisce la creazione di servizi server-side aperti a tutti i richiedenti.

Handlebar.js

- Handlebar.js è una semplice estensione di Mustache.js
- Un linguaggio di template logicless per tenere viste e codice separati
- Permette di accedere agli oggetti annidati dentro alle variabili da interpolare e di ottenere l'effetto di cicli e istruzioni condizionali usando bocchi contestuali.
- https://handlebarsjs.com/

Passport.js (non pre-installato)

- L'autenticazione può venire realizzata da package aggiuntivo di express.
  - Ce ne sono dozzine
  - Alcune autenticazioni base sono incluse direttamente in Express
  - Ad esempio *HTTP basic digest authentication*
- Passport.js è uno dei package più adottati
  - Flessibile e modulare
  - Supporta numerosissimi modelli di autenticazione: HTTP digest, Facebook, Google, Twitter, LinkedIn, ecc., chiamati *strategie*.
  - Permette anche di aggiungere nuove strategie ad-hoc.

nodemon

- Una semplicissima, utilissima utility per node.
- Node.js fa prefetch e cache degli script da eseguire. Quindi ogni modifica sugli script non viene percepita ed usata da node.js fino al riavvio, che va fatto a mano ogni volta.
- In fase di debugging, questo significa che node va riavviato a mano ogni pochi minuti, che può essere sgradevole e pesante.
- Nodemon tiene sotto controllo tutti i file di una directory, e ogni volta che percepisce il salvataggio di una modifica riavvia node.js automaticamente.
- Attenzione!
  - Voi scrivete il file su una delle macchine del laboratorio;
  - Il file system di gocker viene avvisato della modifica dopo 2-3 secondi;
  - Nodemon si accorge della modifica dopo 1 secondo e riavvia node.js;
  - Node.js richiede 3-4 secondi per riavviarsi.
- Nel frattempo il vostro sito darà errore 500 Internal Server Error
- Ci vogliono mai meno di 5 secondi, e a volte anche 10 secondi per rivedere il vostro sito online!!!

MongoDB

- In generale, un'applicazione web, anche data-oriented, non ha bisogno di un DB né SQL né di altro tipo, poiché tutti i linguaggi server-side (da Perl a PHP a node.js) forniscono metodi per utilizzare nativamente il file system e la applicazione può scrivere i suoi dati persistenti su un file qualunque.
- Si usa un DB:
  - quando il file diventa troppo grande per stare tutto in memoria,
  - quando ho bisogno di metodi veloci per cercare un sottoinsieme di dati all'interno del file,
  - quando ho bisogno di gestire accessi concorrenti in scrittura.
- Usare un DB per leggere, ma non modificare, un file intero di dimensioni non enormi è come comprare una Ferrari per andare a far la spesa. Funziona, ma è complicato, ed è uno spreco di tempo e risorse.

MongoDB - Introduzione

- MongoDB è un database NoSQL orientato ai documenti.
- Esso memorizza i documenti in JSON, formato basato su JavaScript e più semplice di XML, ma comunque dotato di una buona espressività.
- I documenti sono raggruppati in collezioni che possono essere anche eterogenee. Ciò significa che non c'è uno schema fisso per i documenti. Tra le collezioni non ci sono relazioni o legami garantiti da MongoDB.
- MongoDB si adatta a molti contesti, in generale quando si manipolano grandi quantità di dati eterogenei e senza uno schema.
- MongoDB fornisce un driver per JavaScript **lato server** ossia per **Node.js**. Sulle macchine di cs è preinstallato, sulle vostre dovrete installarlo di persona.

MongoDB - CRUD

- Un database, è un insieme di collezioni che corrisponde ad un insieme di file nel disco fisso.
- Un'istanza di MongoDB può gestire più database. Di seguito mostreremo brevemente come effettuare le operazioni CRUD (Create, Read, Update, Delete).
- Su MongoDB, non esiste alcun comando esplicito per creare un database.
- Anche le collezioni vengono create implicitamente al primo utilizzo.

MongoDB - CRUD

Per **inserire** un documento:

```
db.websites.insert(
    {
        name: "homepage",
        url: "http://www.html.it",
        tags: ["Development", "Design", "System"]
    }
);
```

La **risposta del server** sarà la seguente:

```
{ "nInserted" : 1 }
```

che dice che è stato inserito un documento nel database.

MongoDB - CRUD

Per **interrogare tutti i documenti presenti nella collezione**:

```
db.website.find()
```

in questo caso il server risponderà così:

```
{
  "_id" : ObjectId("5450e468efaef47248f4412c"),
  "name" : "homepage",
  "url" : "http://www.html.it",
  "tags" : [ "Development", "Design", "System" ]
}
```

MongoDB - CRUD

Il campo _id **identifica univocamente ogni documento ed è obbligatorio**, oltre ad essere unico nell'intera collezione. Se non assegniamo noi un valore a _id (come abbiamo fatto in questo caso), MongoDB genera un **ObjectId** e lo assegna. Un ObjectId è un numero pseudocasuale che ha un tasso di collisione infinitesimo (è cioè molto difficile generarne due uguali).

Si noti anche che la sintassi di ObjectId non rientra nello standard JSON. Ciò perché in realtà MongoDB usa una sua estensione, chiamata **BSON** (**B**inary J**SON**).

MongoDB - CRUD

L’aggiornamento dei documenti avviene tramite il metodo **update**:

```
db.websites.update(
    { name: "homepage"},
    { $set : { clicks: 5403 } },
    {multi: true}
)
```

MongoDB - CRUD

L’eliminazione di un documento dalla collezione viene effettuata per mezzo del metodo **remove**.

Il primo parametro rappresenta il criterio di selezione dei documenti da eliminare:

```
db.websites.remove( { name: "homepage"})
```

MongoDB - Server

Ad esempio, per accedere ad una collection su un database di MongoDB la sintassi è la seguente:

```
var client = require('mongodb').MongoClient;
client.connect("mongodb://site2021XX:[PWD]@mongosite2021XX
?writeConcern=majority",
    async function(error, libri) {
        if(! error) {
            var autori = libri.collection("autori");
            await autori.insert(
                { nome: "Umberto", cognome: "Eco" });
            db.close();
        }
    }
);
```

MongoDB e il progetto

- Bisogna lanciare MongoDB su un docker separato.
- Usate gocker per lanciare MongoDB.

  create mongoDB site2021XX
- Il server risponde:

  Mongodb username: site2021XX - Mongodb password: YYYYYYY
  You can connect your mongodb from your site web using hostname mongo_site2021XX
- Questi tre dati vanno conservati ed utilizzati dallo script su node.js per collegarsi con successo.
- Mongo non è accessibile all'esterno, ma solo alla vostra installazione gocker.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

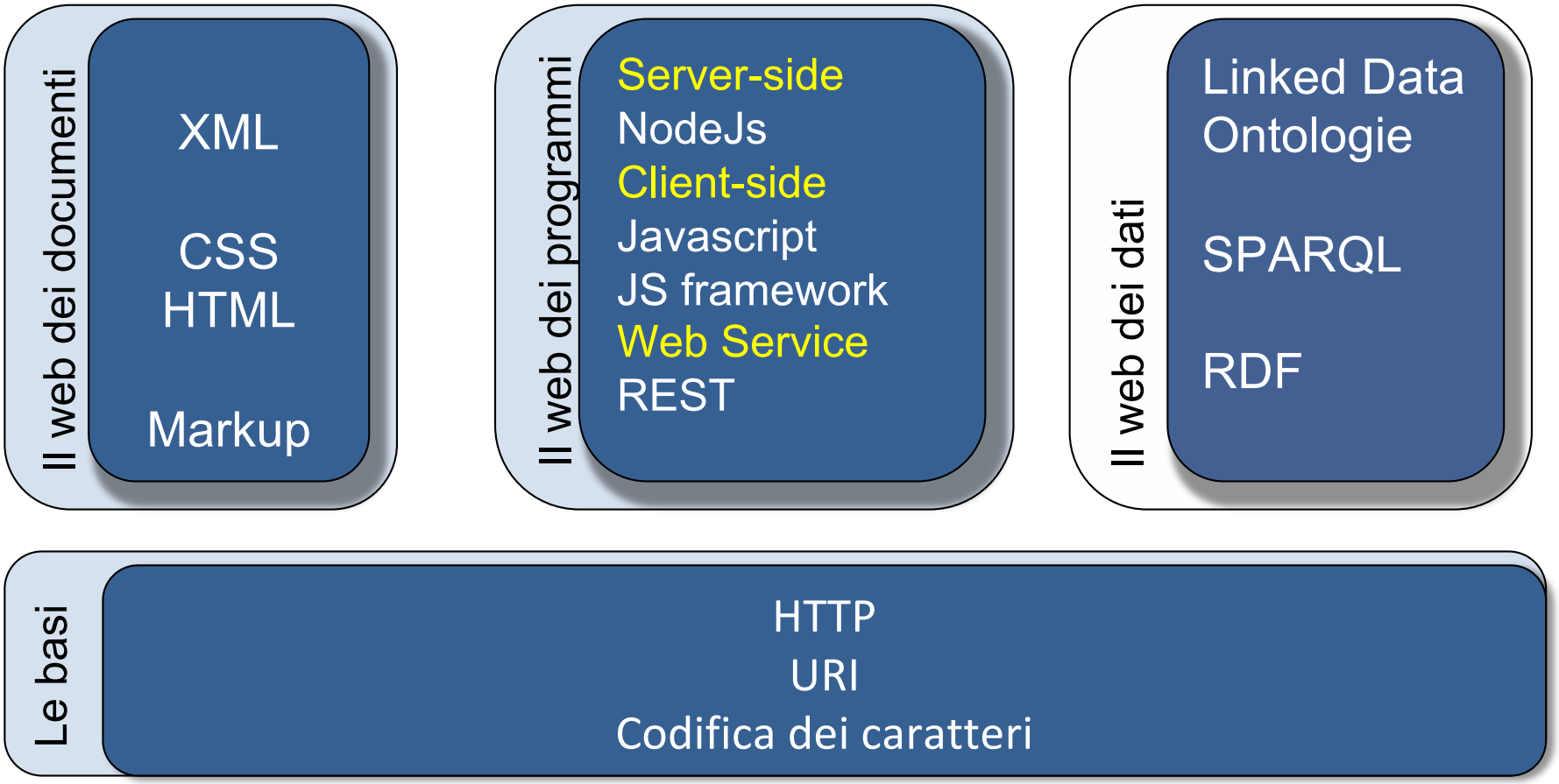
Conclusioni

Di cosa abbiamo parlato
(*e non ci sarà all'esame*)

- Talmud e Bibbia
- Le classificazioni Novarese e Vox-Atypl
- EBCDIC e le code page di ASCII
- Poesia medievale cinese e geroglifici maya ed egiziani
- Pornografia e innovazioni tecnologiche
- Caratteri klingoniani ed elfici
- Vannevar Bush, Douglas Engelbart, Ted Nelson
- Simple Mail Transfer Protocol
- Baudot e le telescriventi del XIX secolo

Di cosa abbiamo parlato

(*e ci sarà all'esame*)



BUON LAVORO!

Fabio Vitali
Francesco Sovrano
Vincenzo Rubano
Angelo di Iorio


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

www.unibo.it