

# Corso Front End Developer

## HTML

Emanuele Galli

[www.linkedin.com/in/egalli/](https://www.linkedin.com/in/egalli/)

# Software Developer

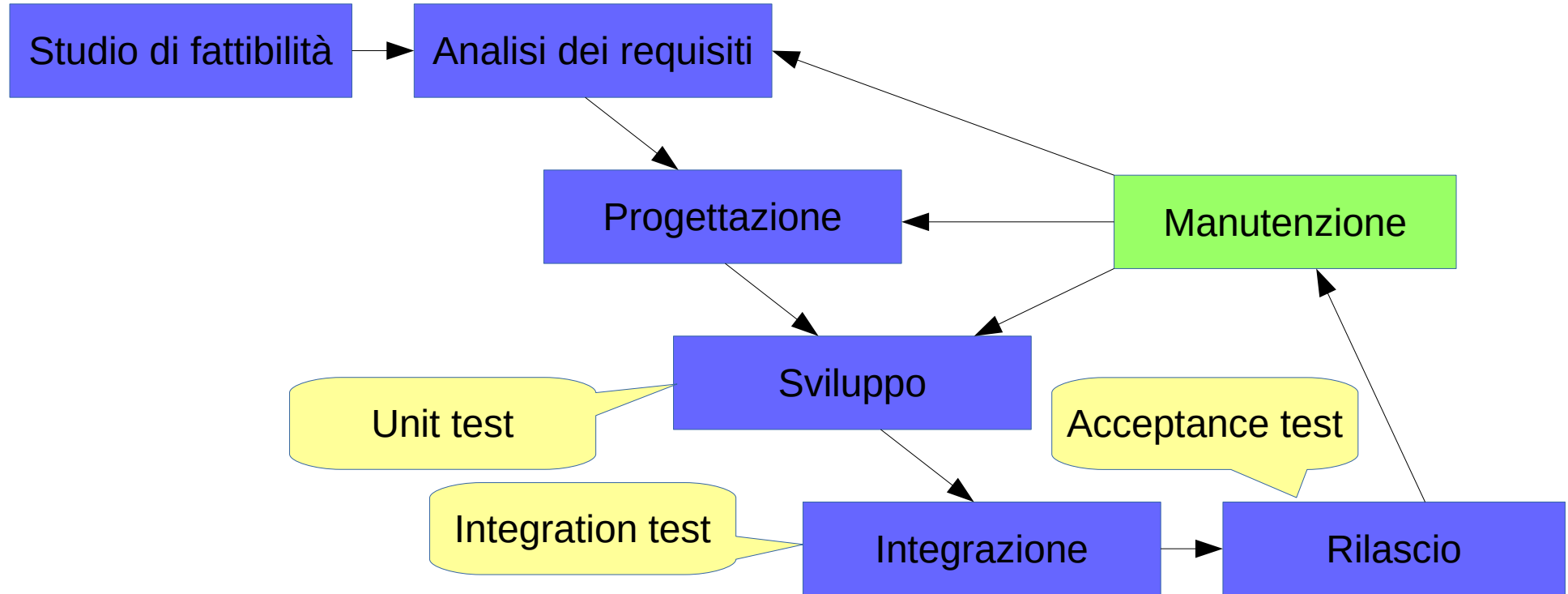
- Front End Developer
  - Pagine web, interazione con l'utente
    - HTML, CSS, JavaScript
    - User Experience (UX)
- Back End Developer
  - Server (controller + database)
    - Java, C/C++, Python, JavaScript, SQL, ...
    - Spring, Node, DBMS, ...
- Full Stack Developer
  - Sintesi (mitologica) delle due figure precedenti

# Ciclo di vita del software

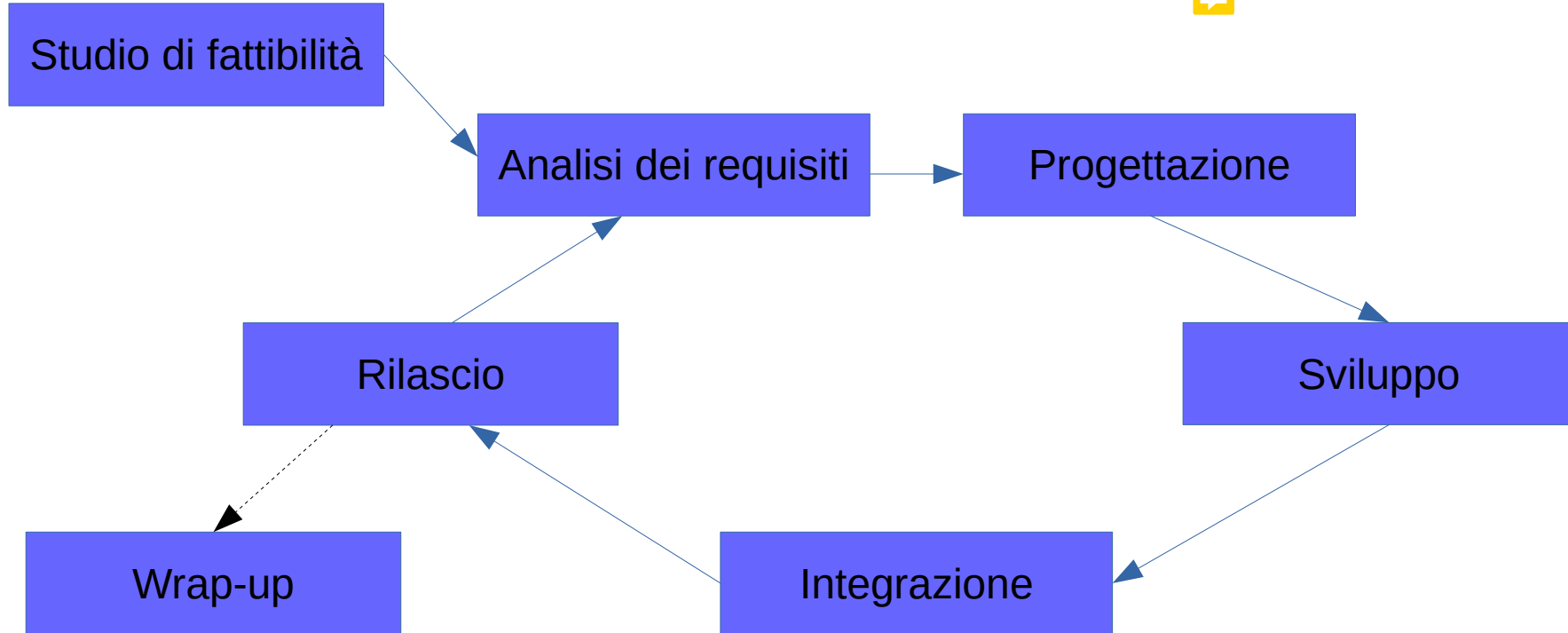
Come gestire la complessità di un progetto?

- Divide et impera
- Struttura
- Documentazione
- Milestones
- Comunicazione e interazione tra partecipanti

# Modello a cascata (waterfall)



# Modello agile





# Atomic Design





- Atomo
  - Elementi di base
- Molecola
  - Più atomi che concorrono ad uno scopo comune
- Organismo
  - Componente complesso di molecole cooperanti
- Template
  - Sorta di pagina astratta, completa ma generica
- Pagina
  - Risultato concreto



# Tool

- Editor di testo
  - vim, notepad, ... 
- Code editor
  - emacs, atom, vs code , ...
- Ambiente di sviluppo integrato (IDE)
  - Eclipse, JetBrains, Visual Studio, ...

# Internet

- Estensione di Arpanet
- Rete di comunicazione basata su TCP/IP 
  - TCP vs UDP
- Nodi identificati da indirizzo IP 
  - DNS: Domain Name System
- Telnet, FTP, ... (identificati da porta) 
- HTTP → World Wide Web 



# Tre tecnologie per il web

 HTML struttura, CSS stile, JavaScript interattività

- Standard del W3C 

<https://www.w3.org/standards/webdesign/>

- MDN Mozilla Developer Network

  
<https://developer.mozilla.org/it/docs/Web>

# HTML: HyperText Markup Language

- Tim Berners-Lee @CERN ~1990
- World Wide Web Consortium (W3C) HTML5 2014
- Descrive come rappresentare pagine web
- Il rendering è responsabilità del browser
  - Chrome
  - Firefox
  - Safari
  - ...
- Struttura ad albero, ogni nodo è un elemento
  - DOM: Document Object Model

```
<!DOCTYPE html>
<!-- my hello page -->
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

hello.html

# Elemento

## Singolo componente di un documento HTML

- Normalmente delimitato da **open** – **close** tag
- Può contenere testo e altri elementi
- Può avere **attributi** nella forma nome="valore"
- "!" indica che non è un elemento
  - **DOCTYPE** tipo di documento. Aiuta il browser a interpretare correttamente il codice (qui: HTML5)
  - Commenti HTML: <!-- ... -->




```
<!DOCTYPE html>
<!-- my hello page -->
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

# head vs body





- **html**
  - Contiene l'intero codice HTML della pagina
- **head**
  - Informazioni *sulla* pagina
- **body**
  - Informazioni *nella* pagina

```
<!DOCTYPE html>
<!-- my hello page -->
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

# head

- Gli elementi in head hanno lo scopo di **descrivere la pagina corrente**
  - **title**: il titolo della pagina, solitamente mostrato dal browser nella barra del titolo  
`<title>Hello</title>`
  -  **meta**: informazioni aggiuntive, come l'encoding usato e indicazioni per i motori di ricerca  
`<meta charset="utf-8">`   
`<meta name="description" content="Writing HTML code">`   
`<meta name="keywords" content="html, head, title, meta">`

# Testo

- h1..h6 
  - Titoli (heading) di parti del testo
- p
  - Paragrafo, unità di base per la suddivisione del testo
- b, i, sup, sub, ... 
  - Formattazione del testo, <b>(bold → grassetto)</b>, <i>(italic → corsivo)</i>, etc
  - Obsoleti (andrebbe usato CSS) ma mantenuti per compatibilità e semplicità
- br 
  - HTML ignora spazi, tab, andate a capo, etc.
  - Per forzare l'andata a capo si usa <br> o <br/>, elemento che  non ha tag di chiusura
- hr
  - Per separare blocchi nella pagina si può usare un horizontal ruler <hr> o <hr/>

# Caratteri speciali

- Alcuni caratteri non utilizzabili in HTML, o non disponibili su normali tastiere, sono resi con “entity”, stringhe che iniziano con “&” e finiscono con “;”

&lt; <

&euro; €

&gt; >

&cent; ¢

&amp; &

&copy; ©

&quot; "

&reg; ®

- <https://dev.w3.org/html5/html-author/charref>




# Liste

- ol
  - Lista ordinata in cui ogni voce ha un indice crescente
  - L'elemento ol contiene un elemento li (list item) per ogni voce
- ul
  - Lista senza ordine, come ol ogni voce è un li, ma pallino (o altro) invece di indice
- dl
  - Lista di definizioni, dl può contenere ogni combinazione di elementi dt e dd
    - dt (definition term), il termine da definire
    - dd (definition of definition), la definizione del termine





# Link

## Gestione dell'ipertestualità nelle pagine HTML

- a – href
  - anchor to an hypertext reference, “ancora” l'elemento ad una risorsa definita nel suo attributo href
  - risorsa interna: `<a href="index.html">index page</a>`
  - elemento nella pagina corrente
    - Definito un elemento con un dato id: `<h1 id="top">Hello</h1>`
    - Un anchor può linkarlo così: `<a href="#top">the top</a>`
  - href a (un elemento in) un risorsa nel web: `https://www.w3.org/#w3c_crumbs` 
  - mail-to: `<a href="mailto:adm@example.com">site administrator</a>`

# Immagini

- **img** – src, alt, title, height, width 
  - *Non ha tag di chiusura*, tutte le informazioni sono negli attributi
  - **src**: l'indirizzo della risorsa, che può essere locale o meno
    - ``
    - ``
  - **alt**: testo alternativo, da mostrare se l'immagine non è accessibile
  - **title**: testo aggiuntivo mostrato quando il puntatore passa sull'immagine
    - ``
  - **height, width**: dimensioni dell'immagine
    - Specificandone una l'altra viene calcolata dal browser. Entrambe: l'immagine può essere distorta
    - Valore assoluto (pixel): ``
    - Percentuale sul viewport corrente: `` 
- **figure** (HTML5) contiene **img** e la descrizione relativa come **figcaption**



# iframe



- Inline frame – permette l'embedding di un'altra pagina HTML in quella corrente
- L'attributo chiave è **src**, generato dal sito ospite

```
<iframe src="https://www.openstreetmap.org/export/embed.html?bbox=9.19%2C45.46%2C9.19%2C45.46">
</iframe>
```

```
<iframe src="http://maps.google.it/maps?q=duomo+milano&output=embed">
</iframe>
```

# Tabelle

- table
  - Tabella descritta come collezione di righe (dall'alto verso il basso), a loro volta descritte come collezione di celle (da sinistra a destra)
- tr
  - Riga nella tabella (table row)
- td
  - Descrive una singola cella (table datum)
  - Attributi **colspan**, **rowspan**
- th
  - Descrive una cella di intestazione
  - L'attributo opzionale **scope** indica se "row" o "col"

```
<table>
  <tr>
    <th></th>
    <th scope="col">Left</th>
    <th scope="col">Right</th>
  </tr>
  <tr>
    <th scope="row">Top</th>
    <td>LT</td><td>RT</td>
  </tr>
  <tr>
    <th scope="row">Bottom</th>
    <td>LB</td><td>RB</td>
  </tr>
</table>
```

Rendering standard: nessun contorno a tabella e celle (CSS)

	<b>Left</b>	<b>Right</b>
<b>Top</b>	LT	RT
<b>Bottom</b>	LB	RB

# Blocco = div, inline = span

- Alcuni elementi implicano la creazione di un nuovo **blocco**, come h1..6, p, ul, li
- Altri, **inline**, sono considerati parte del blocco già esistente, come a, b, img
- L'elemento **div** rappresenta un blocco generico
- La sua controparte inline è **span**
- Sono più flessibili ma non veicolano facilmente il loro senso
- Gli altri elementi sono detti “semantici”
  - Preferiti dai motori di ricerca (vedi SEO: Search Engine Optimization)
  - Semplicità di lettura e comprensione anche nello sviluppo della pagina

# id vs class

- L'attributo **id** permette di identificare **univocamente** un qualunque elemento all'interno di una pagina
- L'attributo **class** permette di identificare un **gruppo** di elementi in un pagina
- L'uso di class e id è fondamentale nell'interazione tra HTML con CSS e JavaScript

# Interazione con utente

- L'elemento **form** è uno tra i principali strumenti per gestire l'interazione con l'utente
- Permettono di inviare dati al sito web
- Il form contiene **widget** (elementi HTML visualizzati in modo standard), ognuno dei quali è usato per generare un parametro con i dati da inviare

# Request – Response

- Il submit di un form genera una request che viene indirizzata al server usando il protocollo HTTP specificando
  - Metodo usato, tipicamente GET o POST
  - URL destinatario
  - Parametri associati, visti come coppie name → value
- Il server gestisce la request e alla fine genera una response che viene ritornata al chiamante
- Il browser mostra il risultato all'utente



# form

- Gli attributi fondamentali di un elemento **form** sono:
  - **action**: URL dove devono essere mandati i dati
  - **method**: quale metodo HTTP deve essere usato per spedire il messaggio (default GET)

```
<form action="/comment" method="post">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="sender">
  </div>
  <div>
    <label for="msg">Message:</label>
    <textarea id="msg" name="message"></textarea>
  </div>
  <div>
    <button type="submit">Send</button>
  </div>
</form>
```

# Submit di un form

- In questo esempio l'input dell'utente avviene via:
  - `input-text` (stringa di testo)
  - `textarea` (blocco di testo)
- L'attributo `name` in ogni widget determina l'associazione con il parametro passato al server
- Le `label` chiariscono il ruolo del widget associato
  - L'attributo `for` collega una label al controllo con quell'`id`
- Il `button-submit` reagisce a un click dell'utente eseguendo l'azione del form

```
<form action="/comment" method="post">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="sender">
  </div>
  <div>
    <label for="msg">Message:</label>
    <textarea id="msg" name="message"></textarea>
  </div>
  <div>
    <button type="submit">Send</button>
  </div>
</form>
```

# input text (et al.) – textarea

- **input**

- *Non ha closing tag*, per assegnare un valore di default si usa l'attributo **value**

L'attributo **placeholder** visualizza una indicazione per l'utente su quello che ci si aspetta come input

- Se è un parametro obbligatorio si può usare la validazione HTML5 con l'attributo **required**
- L'attributo **maxlength** fissa la lunghezza massima del valore
- L'attributo **type** determina il suo tipo specifico, tra cui:

- **text** (default) `<input type="text" name="user" value="Bob" maxlength="30">`
- **password** (dati sensibili) `<input type="password" name="pwd" maxlength="30" required>`
- **hidden** (parametro nascosto) `<input type="hidden" name="invisible" value="notShowed">`
- **date** (scelta di un giorno) `<input type="date" name="milestone">`
- **file**, **email**, **url**, ...

- **textarea**

- Blocco di testo su più righe, tra open e close tag si può inserire il testo di default

`<textarea name="comment">Enter your comment here.</textarea>`

# input radio

- Scelta di una opzione da una lista
- L'attributo `checked` indica la scelta di default
- Al click del submit button, il radio button checked determina quale value viene associato al `name` e messo nella request

```
<input type="radio" id="favJ" name="fav" value="Java" checked>  
<label for="favJ">Java</label>  
<input type="radio" id="favPy" name="fav" value="Python">  
<label for="favPy">Python</label>  
<input type="radio" id="favCpp" name="fav" value="Cpp">  
<label for="favCpp">C++</label>
```

# input checkbox

- Scelta di più opzioni da una lista
- L'attributo `checked` indica le scelte di default
- Al click del submit button, se c'è almeno un checkbox checked, vengono associati al name e messo nella request

```
<input type="checkbox" id="langJ" name="lang" value="Java" checked>  
<label for="langJ">Java</label>  
<input type="checkbox" id="langPy" name="lang" value="Python">  
<label for="langPy">Python</label>  
<input type="checkbox" id="langCpp" name="lang" value="Cpp" checked>  
<label for="langCpp">C++</label>
```

# select – option

- Scelta di una opzione da una lista a scomparsa
  - **select** fa da container e definisce l'attributo **name**
    - Selezione di più opzioni aggiungendo l'attributo **multiple**
  - **option** definisce il **value** per ogni singola scelta
    - L'attributo **selected** specifica il valore di default

```
<select name="os">  
  <option value="none">-</option>  
  <option value="linux" selected>Linux</option>  
  <option value="windows">Windows</option>  
  <option value="macOs">MacOS</option>  
</select>
```

# fieldset

- **fieldset**
  - Permette di raggruppare campi correlati, migliorando la leggibilità di un form
- **legend**
  - Descrive il fieldset corrente

```
<fieldset>  
  <legend>User</legend>  
  <label>First name: <input type="text" name="fname" /></label>  
  <label>Last name: <input type="text" name="lname" /></label>  
</fieldset>
```