

# **GUIDA GALATTICA PER ASPIRANTI WEB DEVELOPER**

Cristian Baldi - *@crisbal*



unix**MiB**

# Introduzione

## Cosa non imparerai:

- A programmare
  - <https://developer.mozilla.org/en-US/docs/Learn/HTML>
    - Ma anche mille altri siti

## Cosa imparerai:

- A districarti nella giungla confusionaria di tecnologie e terminologie legate al mondo della programmazione Web





# Web Development

“Web Development can range from developing a **simple single static page of plain text** to **complex web-based internet applications**, electronic businesses, and **social network** services.”

“A more comprehensive list of tasks to which web development commonly refers, may include **web engineering, web design, web content development, client-side & server-side scripting, web server and network security configuration.**”

Screenshot of a Facebook page for "unixMiB". The page header shows "Page" selected, with "Inbox 1", "Events", "Notifications 4", "Insights", "Publishing To...", and "More" options. The profile picture is a penguin icon, and the name is "Cristian". Below the header, there's a post from "unixMiB" about "BIBLIOHACK DAY 2019". The post text reads: "Ritorna in UniMiB il "BiblioHackDay 2019". Il 15 Marzo, dalle ore 9:00 in Biblioteca U6, tu e il tuo team avrete l'opportunità di creare app e giochi che migliorino l'esperienza nei luoghi di cultura. In palio buoni acquisto negozi e noti store on-line di elettronica. La partecipazione è gratuita (pranzo e RedBull inclusi), ma i posti sono limitati, per registrarsi: <https://goo.gl/forms/k9U7EAuUiYEmE9nl2...> See More". The post has "Liked", "Following", "Share", and "More" buttons. The main content area shows a banner for "BIBLIOHACK DAY 2019" at the Università degli Studi di Milano-Bicocca.

# Mi presento

Ciao a tutti mi presento.

Questo è un [link](#)

```
> curl miosito.it/api/persone.json
[
    {
        "name": "Cristian",
        "age": 22
    },
    {
        "name": "Pluto",
        "age": 3
    }
]
```

# HTML



# Web Browser



# CSS



# JS



# HTML - Hypertext Markup Language

```
<!DOCTYPE html>
<html lang="it">
<head>
  <title>Il mio sito</title>
</head>
<body>
  <h1>Mi presento</h1>
  <p>Ciao a tutti mi presento.</p>
  <p>Questo è un
    <a href="google.it">link</a>
  </p>
</body>
</html>
```

- Linguaggio basato sui tag
- Definisce la struttura e i contenuti del documento
- L'ultima versione è HTML 5

## Mi presento

Ciao a tutti mi presento.

Questo è un [link](#)

# CSS - Cascading Style Sheets

```
body {  
    background-color: pink;  
}
```

```
p {  
    border: 1px dashed black;  
    color: red;  
}
```

- Definisce l'aspetto (stile) della pagina
- L'ultima specifica è CSS 3

**Mi presento**

Ciao a tutti mi presento.

Questo è un [link](#)

# JS - Javascript

```
<button onclick="saluta()">  
    Cliccami  
</button>
```

....

```
function saluta() {  
    var a = 3+2;  
    var b = a/42;  
    alert("Ciao amico!");  
}
```

Cliccami

- Linguaggio di programmazione che permette di rendere la pagina interattiva e dinamica
- Fornisce l'accesso a funzionalità "avanzate" del dispositivo: accelerometro, fotocamera, posizione, file e molto altro

This page says  
Ciao amico!

OK

# Chi scrive l'HTML? (ed il CSS? ed il JS?)

- Uno **sviluppatore front-end**
  - Si occupa principalmente della parte grafica e di interazione utente di un sito web
  - Deve conoscere HTML, CSS e Javascript
  - E' sua responsabilità tutto quello che viene **visualizzato sul browser dell'utente finale**
  - Scrive HTML direttamente oppure tramite Javascript
- Un **programma in esecuzione nel back-end**
  - Al momento della richiesta produce in output dell'HTML che viene visualizzato dal browser
  - Il programma è scritto da uno **sviluppatore back-end**
  - In genere interagisce con risorse esterne al browser dell'utente come ad esempio un database
  - Per siti web dai contenuti dinamici e generati al momento della richiesta

# FRONTEND



# BACKEND

# Scrivere HTML, CSS, JS da “zero”

Bastano un **editor di testo** ed un **browser** (Firefox o Chrome)



Visual Studio Code



# Progetto: il tuo primo sito web

Sito web semplice con poca interazione utente.

- HTML, CSS, Javascript

## Mi presento

Ciao a tutti mi presento.

Questo è un [link](#)

# Scrivere HTML, CSS, JS con un aiuto

## Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

[Get started](#)

[Download](#)

Currently v4.3.1

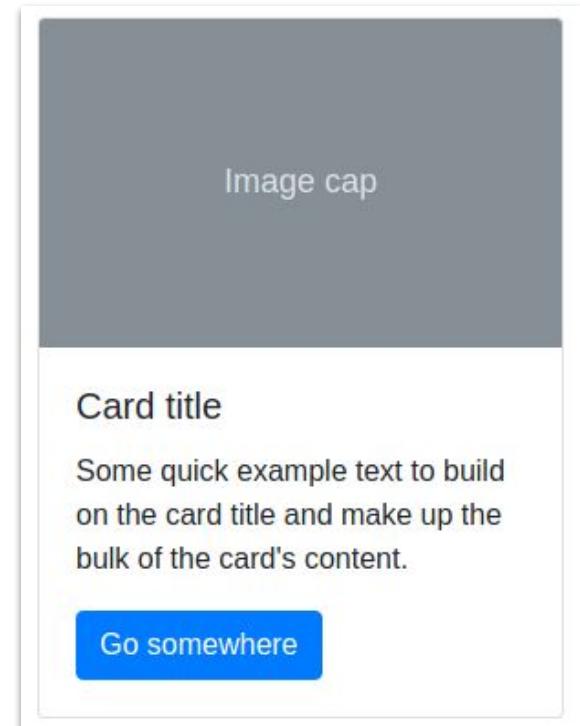


Librerie che offrono CSS  
(e JS) già scritto, testato  
e funzionante per  
**velocizzare lo sviluppo.**

**Sinonimi:** Framework CSS,  
Toolkit Frontend

# Bootstrap: un esempio

```
<div class="card">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text</p>
    <a class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```



# Altri Framework CSS



**Bulma** is a free, open source CSS framework based on **Flexbox** and used by more than **150,000** developers.



33,459

downloads 413k/m



## Foundation

The most advanced responsive front-end framework in the world.

[Download Foundation 6](#)

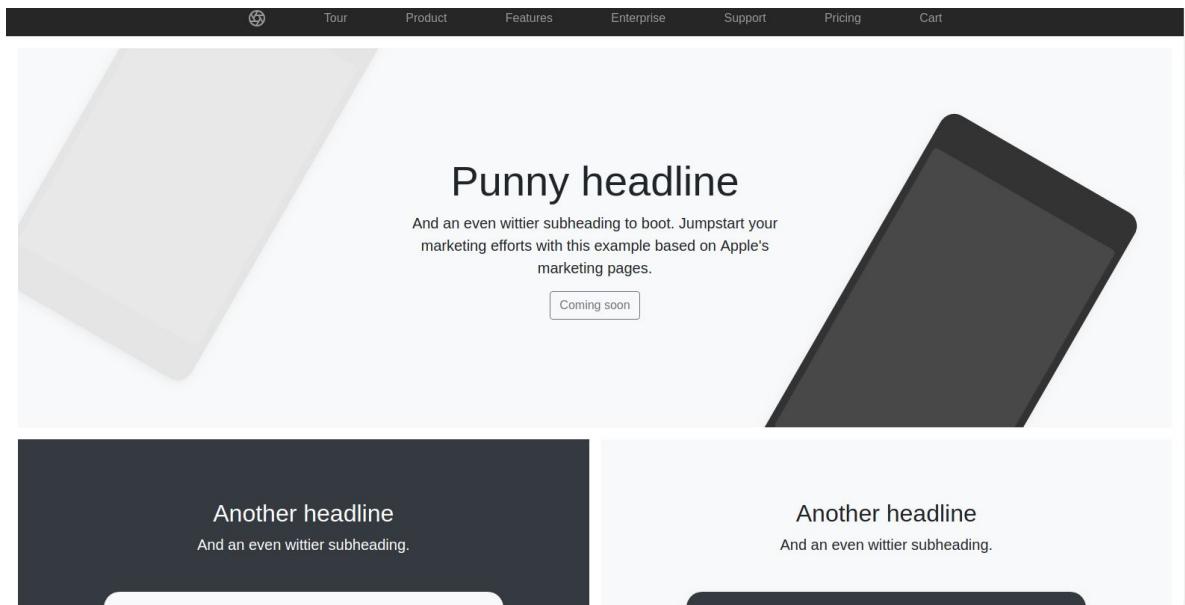
# Parola chiave: responsive



# Progetto: il sito web per il negozio sotto casa

Sito web “vetrina” per una piccola azienda, poca interazione, i contenuti sono importanti e deve vedersi bene da cellulare.

- HTML
- Bootstrap
- Javascript



# Scrivere CSS: pre-processori

- Scrivere CSS “puro” per grandi siti può diventare un’operazione **dispendiosa** in termini di **tempo** e **difficile** in termini di **organizzazione** del codice
  - Non posso riutilizzare facilmente parti di codice
  - Non è semplice scrivere CSS “chiaro”
  - Non posso dichiarare funzioni che modificano i valori
- Ci aiutano dei **linguaggi derivati dal CSS**
  - Estendono il CSS per coprire le sue mancanze
- Il browser capisce solo il CSS: servono dei **preprocessori**/compilatori che a partire dal **linguaggio esteso** si riconducono al CSS

# Scrivere CSS: pre-processori

Sass

{less}

stylus

# Sass: Syntactically Awesome Style Sheets

- E' quello che consiglio e quello che uso
- Due versioni: una più "classica" (SCSS) e una più moderna (SASS, senza punti e virgole e senza parentesi graffe)

```
$myColor: #ff00ff;  
header {  
    color: $myColor;  
    h1.title {  
        color: invert($myColor);  
    }  
}
```

```
header {  
    color: #ff00ff;  
}  
header h1.title {  
    color: #00ff00;  
}
```

# JS Javascript

- Linguaggio di programmazione alto livello ed interpretato
- (Da non confondere con Java!)
- Prima specifica nel 1997, la versione più recente è del 2018
- Ogni browser supporta una versione diversa di Javascript
  - La versione del 2009, ECMAScript 5, è quella supportata da tutti i browser più diffusi
  - Le versioni più recenti introducono funzioni utili e sintassi più espressiva
  - Non si può usare direttamente la versione del 2015 per colpa di Internet Explore 11 (e altri)

## # ECMAScript 5

- OTHER

Usage

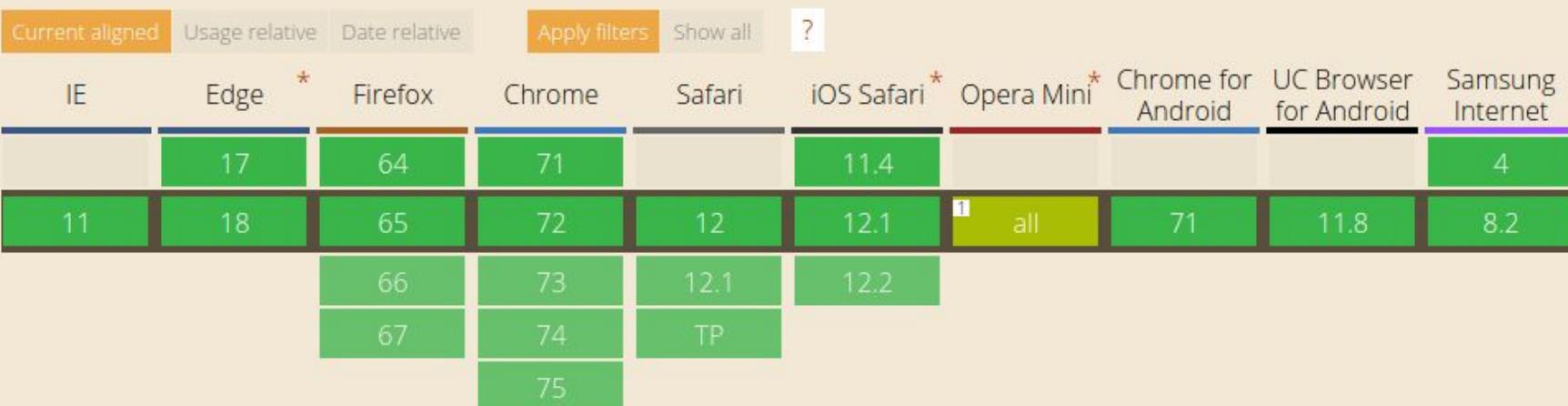
% of all users



Global

92.15% + 2.71% = 94.86%

Full support for the ECMAScript 5 specification. Features include `Function.prototype.bind`, Array methods like `indexOf`, `forEach`, `map` & `filter`, Object methods like `defineProperty`, `create` & `keys`, the `trim` method on Strings and many more.





# JS Moderno su Browser vecchi

Converte in automatico il codice Javascript per mantenere la retrocompatibilità: non tutti i browser supportano le ultime feature di Javascript!

Da:

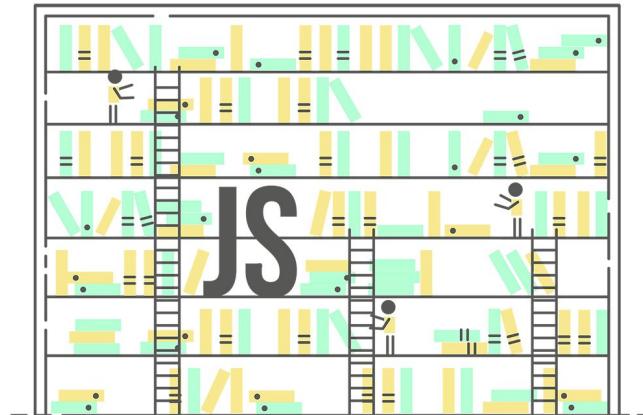
```
[1, 2, 3].map(x => x**2)
```

A:

```
[1, 2, 3].map(function (x) {  
    return Math.pow(x, 2);  
});
```

# Librerie - Lasciamo il lavoro (sporco) agli altri

- Per evitare di riscoprire per l'ennesima volta l'acqua calda
- Codice scritto da altri sviluppatori e reso disponibile (in genere open source)
- Non sono necessarie, si può fare anche senza, però fanno comodo!





- E' una libreria Javascript
  - Niente di più, niente di meno
- Semplifica alcune problematiche comuni
  - Eventi della pagina, accesso a risorse esterne
  - Funziona su tutti i browser
- Si può estendere con plugin/librerie aggiuntive
  - Per fare slider, gallerie di immagini, componenti interattivi



# jQuery? Non è necessario (ma è comodo!)

[YouMightNotNeedJquery.com](http://YouMightNotNeedJquery.com)

## Toggle Class

JQUERY

```
$(el).toggleClass(className);
```

IE9+

```
if (el.classList) {
    el.classList.toggle(className);
} else {
    var classes = el.className.split(' ');
    var existingIndex = classes.indexOf(className);

    if (existingIndex >= 0)
        classes.splice(existingIndex, 1);
    else
        classes.push(className);

    el.className = classes.join(' ');
}
```

## Add a number to another number in JavaScript

hallo

0

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

▼

javascript

★

### 3 Answers

oldest

newest

votes

▲

22

▼

✓

You should definitely use jQuery. It's really great and does all things

answered 11 minutes ago



I<3jQuery

1,234

2 13

[link](#) | [edit](#) | [flag](#)

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – [Isumc0da](#) 8 mins ago

+1 jquery is best quality code ever, if you don't use your a idiot – [Werry\\_Togan](#) 4 mins ago

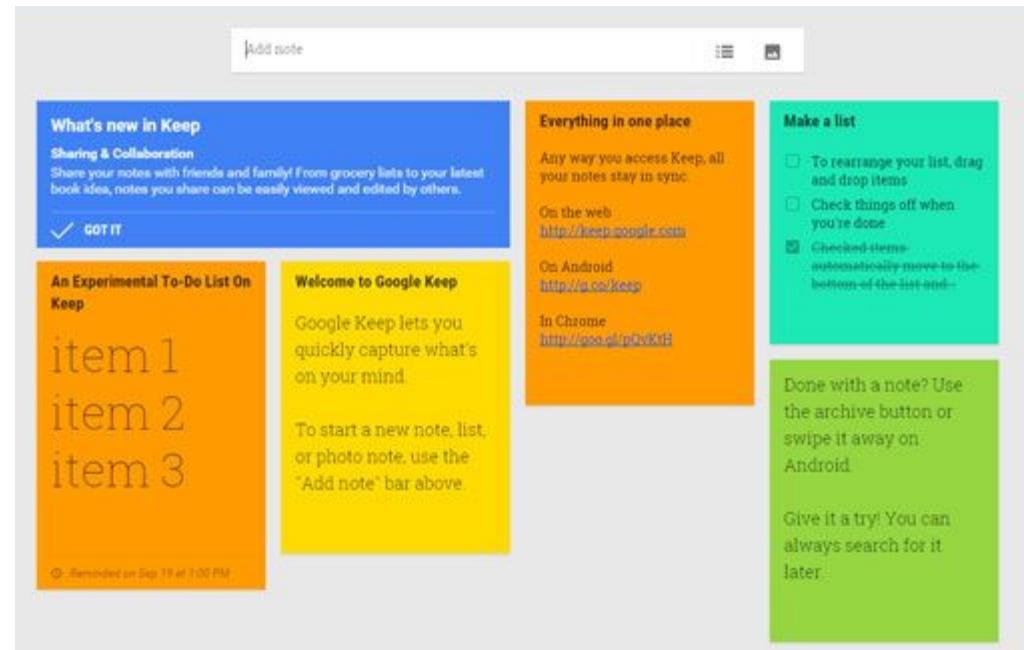
# Altre librerie Javascript

- **Three.js**: per inserire grafica 3D usando WebGL (API 3D per il Web)
- **Axios**: per fare richieste HTTP in modo semplice (e moderno)
- **Chart.js**: per inserire grafici e visualizzazione di dati
- **Moment**: per lavorare con le date in JS senza impazzire
- **Lodash**: per estendere JS con molte funzioni utili non presenti di default
- **Phaser.io**: per scrivere giochi che si eseguono nel browser
- **Reveal.js**: per scrivere presentazioni con HTML
- E centinaia di migliaia di altre!

# Progetto: il blocco note digitale

Una Web App dallo stile personalizzato, responsive, con funzionalità interattive

- HTML
- Sass
- jQuery + Librerie



# Si ma.... come le installo?

Includo lo <script> nella pagina HTML.

```
<script src="https://jquery.com/jquery-3.3.1.min.js"></script>
```

E se ho tanti file?

```
<script src="https://jquery.com/jquery-3.3.1.min.js"></script>
<script src="https://bootstrap.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
<script src="https://momentjs.com/downloads/moment.js"></script>
<script src="https://threejs.org/downloads/three.js"></script>
<script src="https://kenwheeler.github.io/slick/v1.8.1.js"></script>
```

E se ne ho tantissimi? Ci sono soluzioni migliori

# Package Manager - Gestire le dipendenze

- **Dipendenza:** una risorsa (CSS, JS, immagine) necessaria al funzionamento di un progetto
- Gestire le dipendenze è un lavoro **complesso:** è sempre più difficile man mano che si aggiungono dipendenze
- Le dipendenze a loro volta possono avere altre dipendenze
  - Se voglio usare uno script per fare uno slideshow potrei aver bisogno di jQuery
- I siti moderni hanno anche **centinaia di dipendenze**



# C'era una volta: Bower



- Mantenuto...
- ...ma fortemente sconsigliato anche dagli sviluppatori
- Il “primo” package manager per il Web ad essere diventato popolare



# Package Manager - npm e yarn



- Fanno la stessa cosa: gestiscono pacchetti (librerie CSS e JS)
- npm :
  - Il più popolare e lo standard
  - Più stabile e testato
  - Il primo package manager JS
- yarn:
  - Pack. Man. alternativo di Facebook
  - Più recente
  - Era molto più veloce di npm, ora performance simili
- Quale uso? npm va benissimo

# Un problema comune



Librerie Javascript +

Librerie CSS +

Mio codice Javascript +

Mio codice CSS/Sass +

Mio HTML



**Come unisco ed utilizzo tutto insieme?**

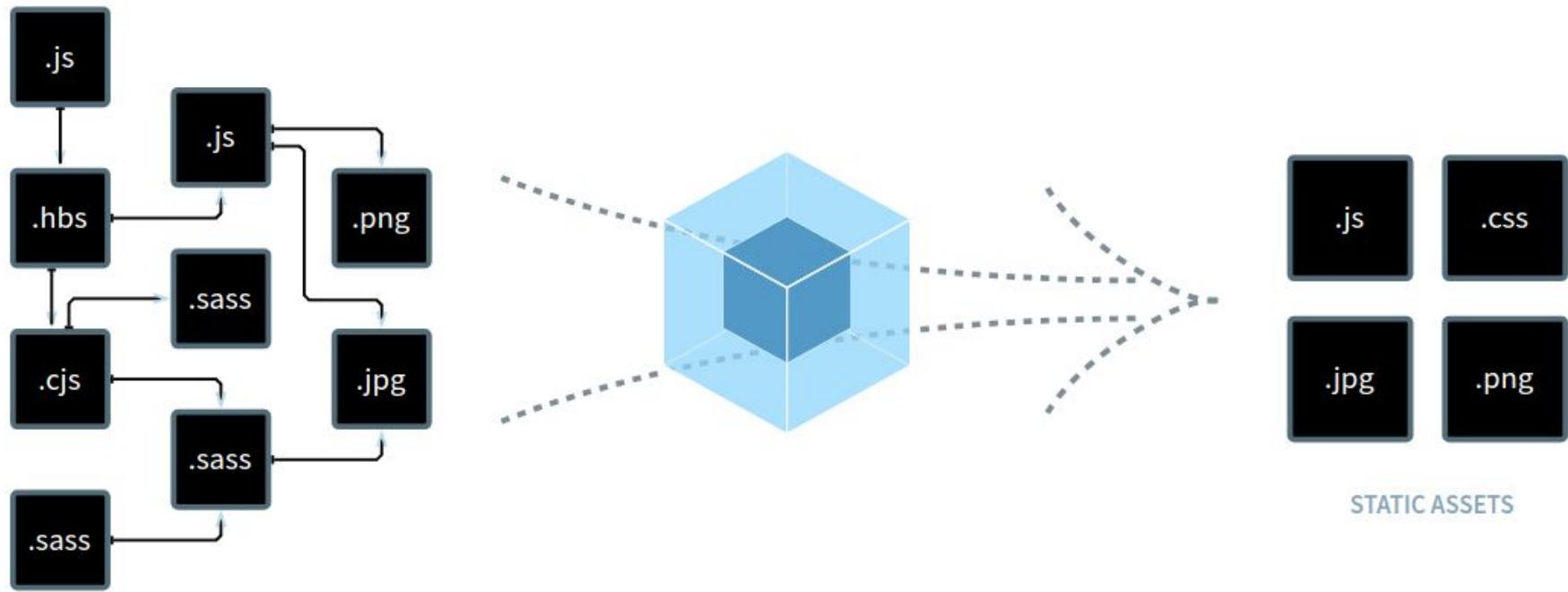
# C'era una volta: gulp e grunt

**Task Runner:** eseguono delle operazioni in modo automatico



- Copia e concatenazione di file
- Compilazione Sass
- Compressione immagini
- Minificazione del codice Javascript
- Auto-reload del browser al cambiamento del codice

# Bundler: webpack



# Bundler: webpack

- Da tanti file diversi ne creo pochi (a volte anche solo uno)
- Fa la stesse cose di Gulp/Grunt ma è un tool specifico
- Meglio di Gulp/Grunt su **progetti complessi**
- **Configurazione non immediata...** ma poi ci si abitua
- Utile per:
  - Supporto a browser vecchi (CSS e JS)
  - Compilazione real time del codice
  - Auto-refresh del browser
  - Compilazione Sass/Less
  - Ottimizzazione immagini
- Esistono moltissimi **plugin** per fare di tutto



# Bundler: parcel



# PARCEL

Blazing fast, zero configuration web application bundler

- Come webpack ma non serve scrivere una configurazione iniziale
- Più veloce nella “compilazione”

## Benchmarks

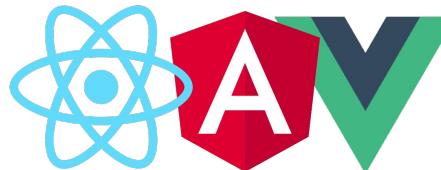
Bundler	Time
browserify	22.98s
webpack	20.71s
<b>parcel</b>	<b>9.98s</b>
<b>parcel - with cache</b>	<b>2.64s</b>

E se voglio fare “Facebook ma più bello”™?



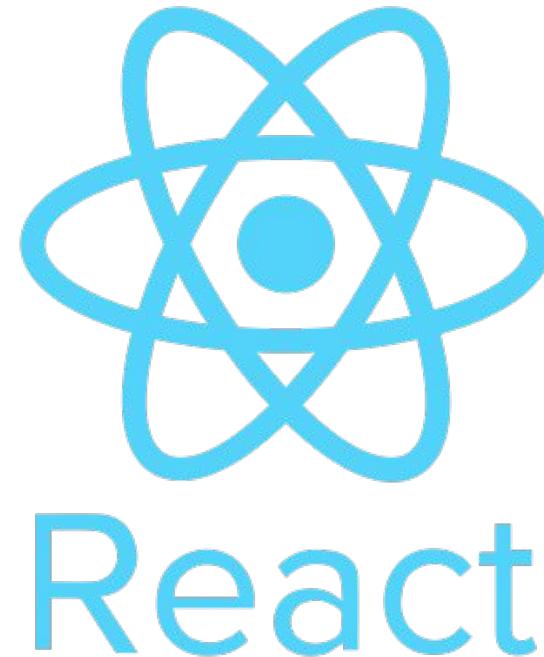
# Framework Frontend: React, Vue, Angular, ...

- Si tratta sempre di Javascript
- Single page application: web app che “girano” in una singola pagina HTML
  - Modificano l'HTML a runtime
- Per applicazioni web altamente interattive e complesse
  - Quando si ha tanta “business-logic” lato utente: workflow complessi, validazione
  - Quando si deve gestire tanti dati dinamicamente: messaggi in real time, post/commenti, data tables, ...
  - Quando si raggiungono “i limiti” con jQuery e Javascript:
    - Gestire il codice e i “componenti” non è più semplice
    - Si reinventano gli stessi concetti dei framework frontend esistenti



# Framework Frontend: React

- Sviluppato da **Facebook**
- Il più **popolare** sia per chi programma che per chi offre lavoro
- React Native: per fare applicazioni native iOS e Android programmando in Javascript
- Sintassi **JSX** per scrivere l'HTML



# React: Sintassi JSX

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

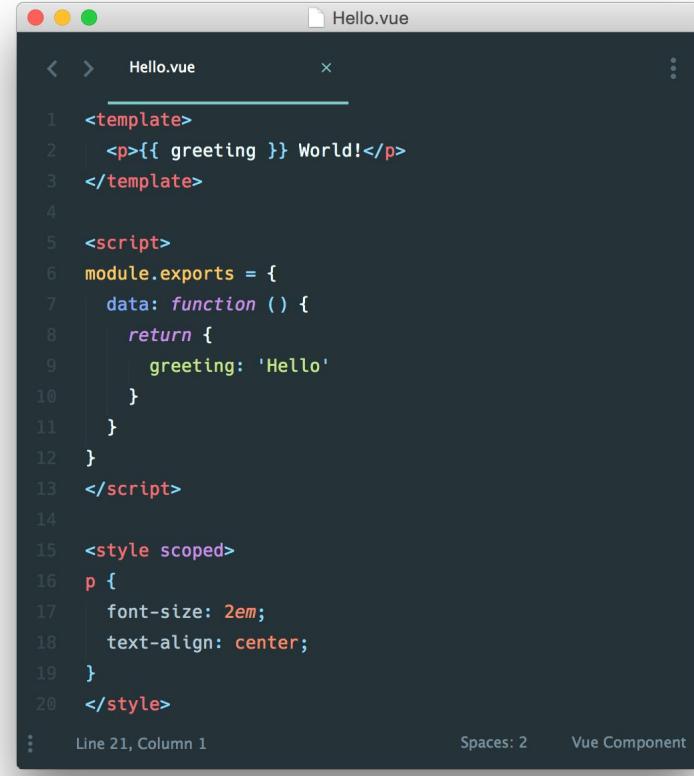
ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

# ♥ Vue.js ♥

- Più **facile** da imparare di React
- Sintassi più **pulita**
- Il mio preferito
- **Single File Components:** tengo tutta struttura, logica e stile di un componente in un solo file



# Vue: Single File Component



A screenshot of a code editor window titled "Hello.vue". The code is a Single File Component (SFC) with the following structure:

```
<template>
<p>{{ greeting }} World!</p>
</template>

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

<style scoped>
p {
  font-size: 2em;
  text-align: center;
}
</style>
```

The code editor interface includes standard window controls (red, yellow, green buttons), a title bar, and a status bar at the bottom indicating "Line 21, Column 1", "Spaces: 2", and "Vue Component".

# Angular

- Sviluppa da **Google**
- Il più “storico”: prima release nel **2010**
- Sintassi semplice per l’HTML
- Non immediato il Javascript: usa **TypeScript**
- Versione legacy: Angular.js



# Quale scelgo?

- **Non è importante**, l'importante è apprendere i concetti principali
- Una volta imparato uno sarà **semplice spostarsi sugli altri**
  - I concetti principali sono gli stessi
  - Ogni framework aggiunge altri concetti o metodologie alla base
- Io consiglio: **Vue.js**
- Il mondo del lavoro consiglia: React
  - Anche se Vue.js sta pian piano prendendo piede



# Extra: React Native

## Build native mobile apps using JavaScript and React

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI using declarative components.

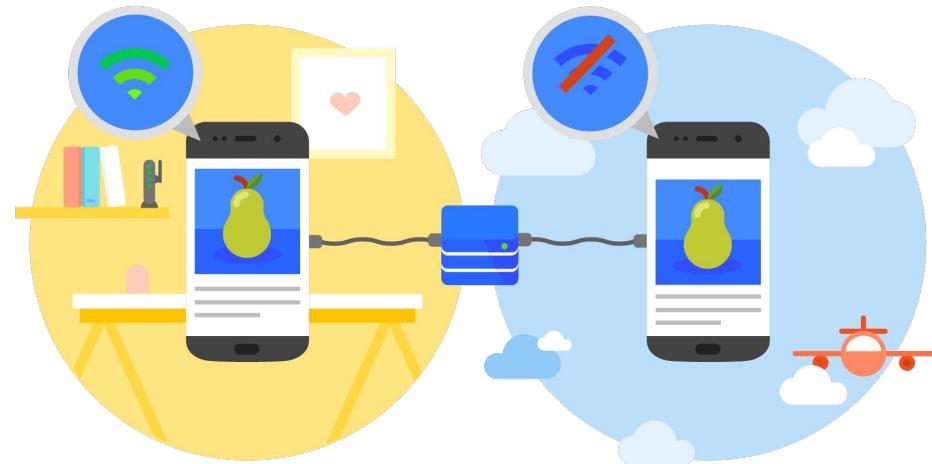
```
import React, {Component} from 'react';
import {Text, View} from 'react-native';

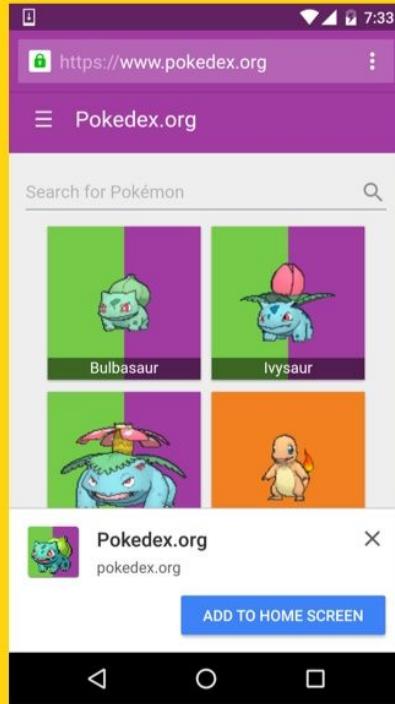
class HelloReactNative extends Component {
  render() {
    return (
      <View>
        <Text>
          If you like React, you'll also like React Native.
        </Text>
        <Text>
          Instead of 'div' and 'span', you'll use native components
          like 'View' and 'Text'.
        </Text>
      </View>
    );
  }
}
```

# Progressive Web App: Web App ~Native

Le PWA sono web app che:

- Funzionano **offline**
- Una volta installate si comportano come **app native**
- Possono inviare **notifiche push** all'utente
- Si **aggiornano** in automatico
- Servite via **HTTPS**

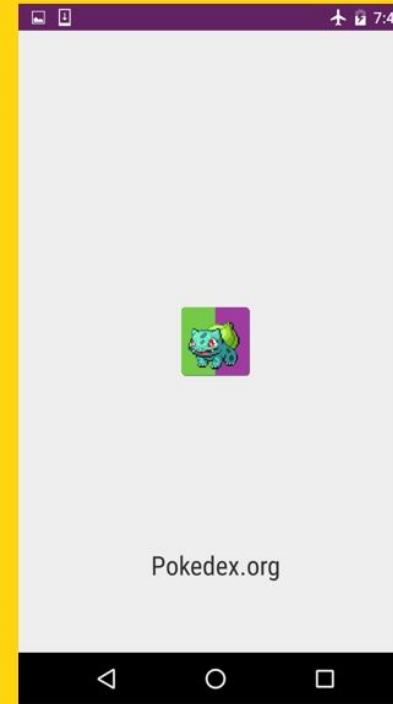




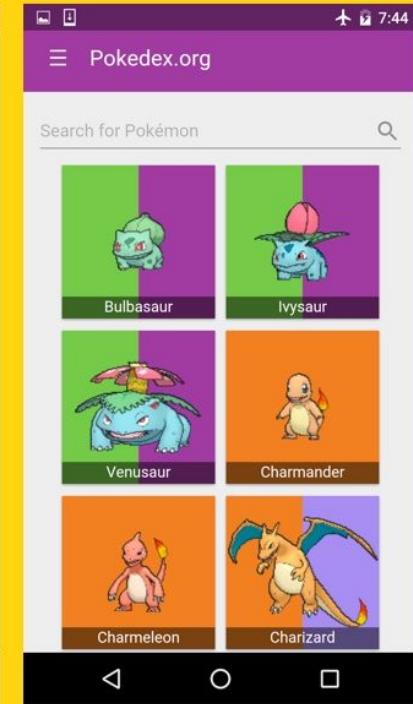
Web App install  
banner for engagement



Launch from user's  
home screen



Splash screen  
(Chrome for Android 47+)



Works offline with  
Service Worker

# Progressive Web App – Manifest

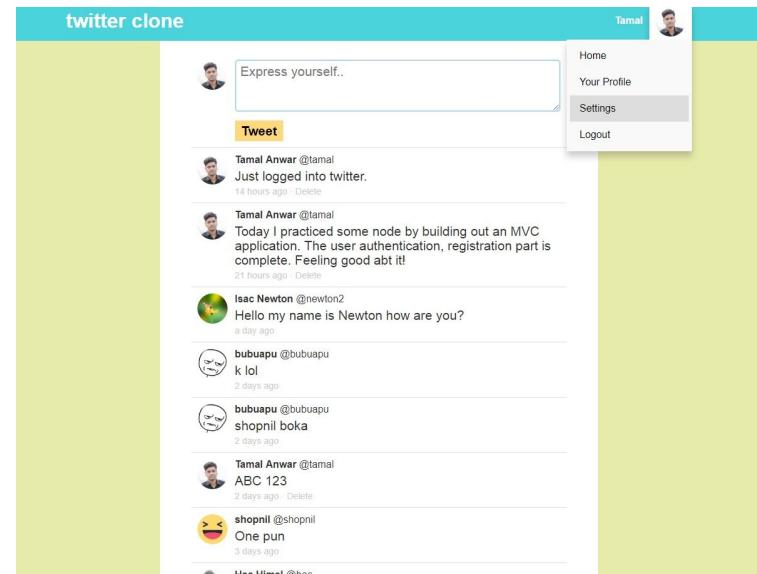
A complete `manifest.json` file for a progressive web app.

```
{
  "short_name": "Maps",
  "name": "Google Maps",
  "icons": [
    {
      "src": "/images/icons-192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/images/icons-512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/maps/?source=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/maps/",
  "theme_color": "#3367D6"
}
```

# Progetto: “Facebook ma più bello”™

Web Application altamente interattiva, dallo stile personalizzato, workflow complesso, che deve offrire delle funzionalità anche offline.

- React/Vue.js
- Sass
- Build: Webpack
- Progressive Web App



# Cosa e come studiare?

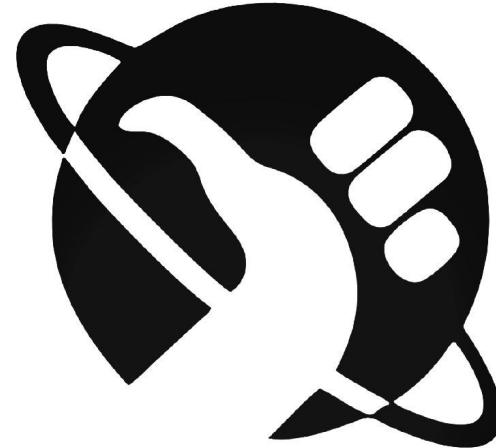
1. Imparare **BENE** HTML (e CSS se si vogliono fare i siti belli da vedere)
2. Imparare Javascript
  - a. Il linguaggio prima, l'interazione con il browser dopo
  - b. Argomenti più difficili: callback, promises, async/await
3. Provare Bootstrap/Bulma
4. Dare un occhio a jQuery ed ad altre librerie
  - a. Imparare a leggere documentazioni di diversi progetti
5. Sass: imparare le basi del linguaggio
6. **Sperimentare** con Webpack: Babel, minificazione, autoprefixing
7. **Sperimentare** con React/Vue.js/Angular

Ma soprattutto

Fare pratica, tanta pratica

# Cosa non abbiamo detto?

- **API**
  - REST e GraphQL, JSON e XML
- **Testing automatico**
  - Testare a mano una web app è lento
- **Accessibilità**
  - Siti Web per tutti gli utenti
- **WebAssembly**
- **Deployment e server HTTP**



**DON'T  
PANIC**

***GRAZIE!***



unix**MiB**