





Meteo

"Che codice fa oggi?"



Smart Meteo

Creiamo una web app per le previsioni meteo sfruttando una API esterna, la geolocalizzazione e l'intelligenza artificiale!

Contenuti:

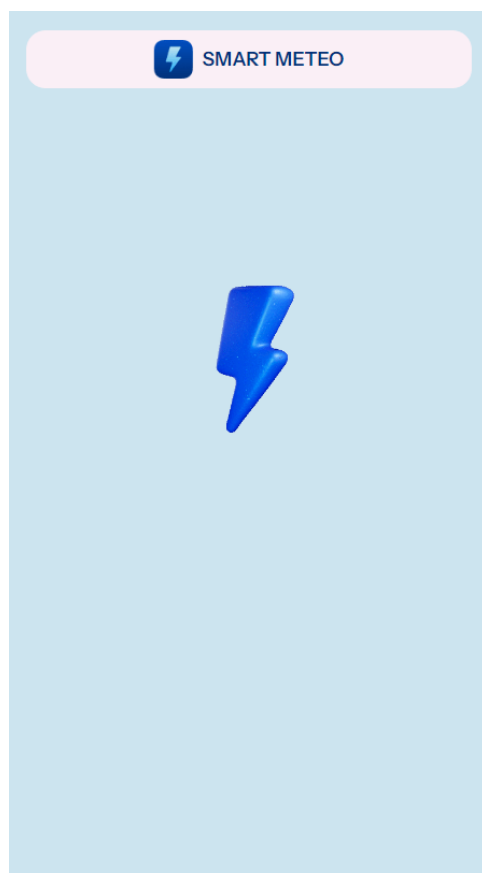
- immagini in background
- linear gradient
- query string
- geolocation
- Open Weather API
- Dall-E API

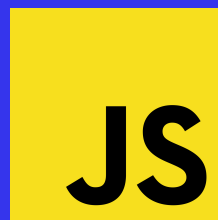




Smart Meteo

Ecco le due schermate che serviranno alla nostra app





JavaScript

le funzioni, più nel dettaglio



Named Functions

Come si scrive una funzione completa?

- keyword:
function
- Nome funzione:
si imposta il nome della funzione con la quale potremo poi richiamarla.
- Codice da eseguire:
Tra le parentesi graffe inseriamo il codice che vogliamo eseguire.
- Parametri: dati in ingresso che possono essere elaborati per restituire un risultato.
- Valore restituito: il risultato di una operazione.

```
1 function miaFunzione(num1, num2) {  
2  
3   // blocco di codice  
4   const risultato = num1 + num2;  
5  
6   return risultato;  
7  
8   // eventuali altre istruzioni dopo il return  
9   // non verranno eseguite  
10  
11 }
```



Non è obbligatorio avere dei parametri, né tantomeno restituire un risultato.



JS

Le Funzioni

Dichiarare e invocare una funzione

Dichiarare una funzione consiste nel definire il suo nome, ciò di cui ha bisogno e cosa deve fare.

Invocare una funzione consiste nel:
scrivere il nome seguito dalle parentesi tonde
nel punto di codice in cui vogliamo usarla.

Una volta invocata, la funzione **eseguirà il codice** in essa contenuto.

```
1 // INVOCAZIONE
2 //
3 nomeFunzione(); //Senza argomento
4
5 nomeFunzione('marco'); //Con argomento
```



QUERY STRING



Query string

Passare al server dati

Un modo per passare dati al server è attraverso l'URI aggiungendo al termine una **querystring**

querystring

www.boolean.careers?email=fabio@boolean.careers&title=CEO



Query string

Passare al server dati

`www.boolean.careers?email=fabio@boolean.careers&title=CEO`

- 1 ? indica che inizia la query string
- 2 chiave=valore
- 3 ogni chiave-valore è separata da un &



JavaScript

Template literal



JS

I template literal

Usare una stringa per creare HTML dinamico

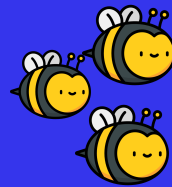
Usando la proprietà `innerHTML` è possibile inserire dell'HTML dinamico all'interno di un nodo del DOM.

```
1 function displayWeatherImage(weatherImageData) {  
2   const src = `data:image/png;base64,${weatherImageData.data[0].b64_json}`;  
3   const img = `![Immagine meteo.](${src})`;  
4  
5   weatherImage.innerHTML = img;  
6 }
```



API

Application Programming Interface





DALL-E 2

Generative AI per le immagini:

- generare immagini da descrizioni testuali
- modificare immagini da istruzioni testuali

Modelli famosi sono DALL-E e Stable Diffusion.



An astronaut lounging in a tropical resort in space in a vaporwave style.

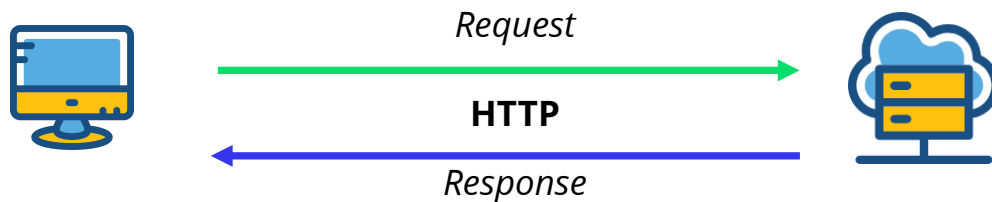


Dall-e Web API

Servizio di creazione e modifica immagini con intelligenza artificiale

Fornendo una descrizione testuale, possiamo chiedere di generare un'immagine di nostro gradimento.

<https://platform.openai.com/docs/api-reference/images/create>



The screenshot displays the OpenAI Platform API reference page for the 'Create image' endpoint. The page is dark-themed and includes a sidebar with navigation links such as Chat, Embeddings, Fine-tuning, Batch, Files, Uploads, Images, Models, Moderations, and a 'REALTIME Beta' section. The 'Images' section is expanded, showing 'Create image' as the selected option. The main content area provides details for the 'Create image' endpoint, including the HTTP method (POST), the URL (https://api.openai.com/v1/images/generations), and a brief description: 'Creates an image given a prompt.' It also defines the 'Request body' with parameters like 'prompt' (required), 'model' (optional, defaulting to 'dall-e-3'), and 'n' (optional, defaulting to 1). An 'Example request' is shown using curl, and a 'Response' example is provided in JSON format, showing a 'created' timestamp and a list of image URLs.



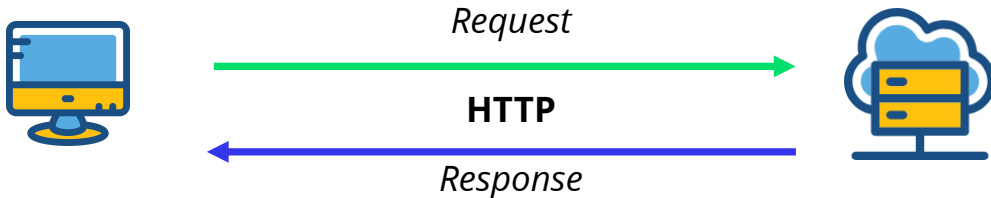
OpenWeather API

Servizio di meteo in tempo reale

Un'API che ci permette di conoscere il tempo meteorologico a seconda di dove ci troviamo!

Sulla documentazione possiamo trovare i parametri corretti per ottenere le informazioni che cerchiamo

<https://openweathermap.org/current>



The screenshot displays the OpenWeather API documentation page for 'Current weather data'. The page includes a navigation bar with links like 'Weather in your city', 'Guide', 'API', 'Dashboard', 'Marketplace', 'Pricing', 'Maps', 'Our Initiatives', 'Partners', 'Blog', and 'For Business'. The main content area is titled 'Current weather data' and includes a breadcrumb trail 'Home / API / Current weather'. The page is divided into two columns. The left column contains sections: 'Product concept' (describing data sources and formats), 'Call current weather data' (with a sub-section 'How to make an API call' showing an example URL), and 'Parameters'. The right column contains a sidebar with links to various API features and documentation topics, such as 'Product concept', 'Call current weather data', 'API response', 'Bulk downloading', 'Other features', 'Geocoding API', 'Built-in geocoding', 'Format', 'Units of measurement', 'Multilingual support', and 'Call back function for JavaScript code'.



HTTP - Facciamo una richiesta

Cosa dobbiamo specificare?

Endpoint

Un'indirizzo da chiamare (sotto forma di *stringa*)

Metodo

Il metodo HTTP con cui effettuare la chiamata (come stringa)

Headers (intestazioni)

Ad esempio il tipo di applicazione, il linguaggio, credenziali, (sotto forma di *oggetto*)

Body

Le informazioni che vogliamo inviare al server (come stringa JSON)

```
1  const endpoint = 'http://example.com/endpoint';
2
3  const response = await fetch(
4    endpoint,
5    {
6      method: '...',
7      headers: { ... },
8      body: JSON.stringify({ ... })
9    }
10 )
```



CSS

Gestire lo sfondo



Position

La proprietà **position: absolute** ci permette di posizionare un elemento al di fuori del flusso orizzontale o verticale del documento rispetto al suo contenitore più vicino che ha una posizione diversa da **static**.

Usiamo **position: relative** per indicare il contenitore che prendiamo a riferimento senza spostarlo. Poi con le proprietà **left | top | bottom | right** controlliamo la posizione dell'elemento.

La proprietà **z-index** indica la posizione sull'asse perpendicolare allo schermo.

```
1 .reference-container {  
2   position: relative;  
3 }  
4  
5 .child-container {  
6   position: absolute;  
7   top: 0;  
8   left: 0;  
9   z-index: -1;  
10 }
```



Adattare l'immagine

Controlliamo le caratteristiche dell'immagine che usiamo di sfondo:

- impostando **height** e **width** a 100% adattiamo l'immagine al suo contenitore
- con **object-fit: cover** ci assicuriamo che l'immagine occupi tutto lo spazio senza perdere le proporzioni originali.

```
1 .image-container img {  
2   width: 100%;  
3   height: 100%;  
4   object-fit: cover;  
5 }
```



Immagine sfumata

Possiamo sfumare un'immagine applicando un gradiente come maschera:

- una *maschera* è un'immagine applicata su un'altra immagine in modo che le parti opache della maschera siano visibili e quelle trasparenti vengano nascoste
- **linear-gradient()** crea un'immagine sfumata da un colore all'altro
- è possibile indicare da dove a dove generare la sfumatura
- con **mask-image** possiamo usare un gradiente da opaco a trasparente per sfumare la nostra immagine

```
1 img {  
2   mask-image: linear-gradient(black 50%, transparent 95%);  
3 }
```



Deploy

Mettiamo l'app online

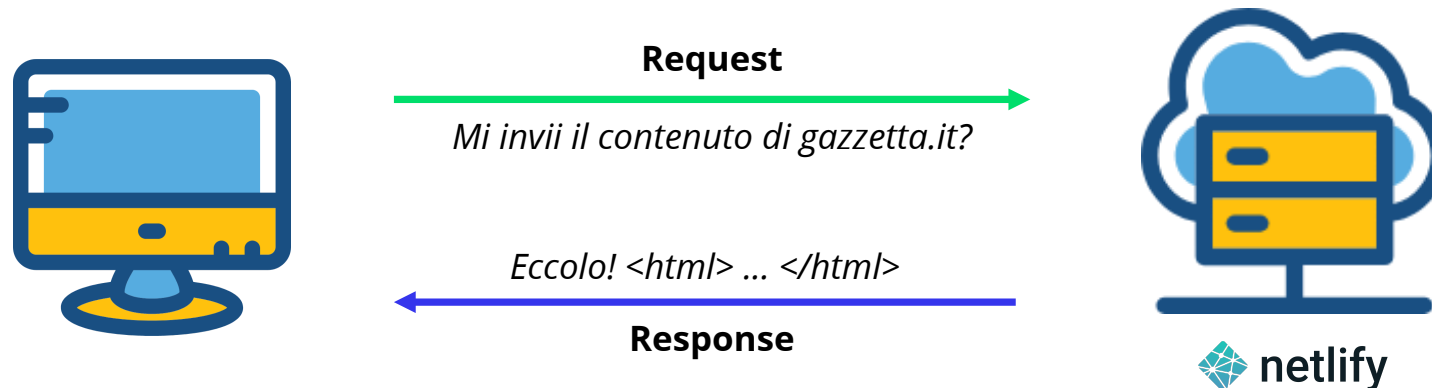


Deploy

Mettiamo il nostro codice su un server.

Sfruttiamo un servizio gratuito che ci consente di trasferire agevolmente il nostro codice e generare un link pubblico per condividere la nostra web app.

<https://app.netlify.com/drop/>





Deploy

1. Registrati Netlify, in questo modo il tuo progetto sarà pubblicato per un tempo illimitato e non richiederà una password
2. Trascina la cartella del tuo progetto dentro Netlify drop
3. Ottieni il link generato e condivisibile

E' fatta!

<https://app.netlify.com/drop/>

