Cos'è CORS?
CORS sta per **Cross Origin HTTP Request** e una web application con dominio X non può richiedere dati ad un dominio Y tramite AJAX se non ha abilitato il CORS. Un browser permette agli script in una pagina web di accedere ai dati contenuti in un'altra pagina web solo se hanno la stessa origine.
Viene implementato inviando degli header HTTP in req/res. Ci sono le richieste semplici e le richieste preflight. Per le richieste semplici sono ammessi i metodi: GET, HEAD, POST, sono ammessi gli header: accept, accept-language, content-language, content-type alcuni valori per l'header content-type.
Nelle richieste preflight fanno predecere alla richiesta principale una richiesta preventiva che si chiama OPTIONS (per chiedere al server quali metodi supporta per il file in questione), quindi in totale si fanno 2 richieste.

Come fare se vogliamo espressamente permettere il resource sharing tra due siti diversi? CORS

Metodi HTTP
I metodi HTTP più comuni sono GET e POST. Il metodo GET è usato per ottenere il contenuto della risorsa che viene indicata nell'endpoint. Mentre il metodo POST è usato per inviare informazioni al server, ad esempio i dati di un form.

Quali metodi HTTP ti ricordi?
GET, POST, PUT, DELETE, PATCH.

Cosa si intende per semantica degli elementi HTTP?
L'HTML semantico si riferisce a un codice HTML che utilizza i tag HTML. E' un modo efficace per descrivere l'obiettivo dei diversi elementi della pagina Un codice HTML semantico trasmette il significato di ogni elemento e segue gli standard così che chiunque possa capire il codice scritto.

Parlami delle API cosa fai dal punto di vista del server per fare richieste agli altri siti?
Le API (Application Programming Interfaces) sono interfacce che permettono alle applicazioni di parlare con altre applicazioni.

Serif e sans serif?
Testo con grazie e senza grazie

Sapresti fare un middleware, che callback gli passi?
Una funzionalità di Express è rappresentata dai **Middleware**. Si tratta di semplici funzioni che possono essere usate all'interno dell'applicazione. Le funzioni Middleware hanno accesso all'oggetto richiesta (solitamente denominato req), all'oggetto risposta (res) e a una funzione callback che viene solitamente denominata next. Le funzioni Middleware sono solitamente disposte in cascata. Per fare in modo che una passi il controllo alla funzione successiva è necessario invocare la funzione next(). Solitamente, se non viene inviata una risposta al client, si invoca la funzione next() per fare in modo che le altre funzioni Middleware lungo la catena possano elaborare la richiesta e passarla eventualmente alla funzione successiva.

Quali campi form conosci?
Nel tag input si possono mettere come type : image, checkbox, radio, submit, reset, text.
Poi c'è il tag SELECT con le varie OPTION VALUE.

Caratteristiche di Javascript?

- Dynamic = gira su una macchina virtuale e non è compilato
- Case sensitive = attenzione alle maiuscole
- Loosely typed = non c'è il bisogno di specificare il tipo delle variabili

La specificità del css?
La specificità è una serie numerica che permette ai browser di capire quale regola CSS ha la priorità sulle altre che cercano di selezionare lo stesso elemento. Prende il controllo la proprietà più vicina all'oggetto.

Ad ogni dichiarazione è attribuita una specificità misurata con quattro valori [a,b,c,d]. Dove "a" è il valore più importante e "d" il meno importante. Si confrontano i valori più importanti e se uguali si passa a quelli successivi altrimenti termina

Cosa sono i web font?
I web font sono font esterni importati con la regola @font-face {
font-family: myfirstFont;

Font-family
Una famiglia di caratteri è un insieme di caratteri che hanno un design comune. I caratteri all'interno di una famiglia, tuttavia, differiscono l'uno dall'altro nello stile come il peso (leggero, normale, grassetto, semi-grassetto, ecc.).

Tutti i font cominciano con la lettera maiuscola tranne quelli generici.
Se il nome contiene uno spazio allora va messo tra le virgolette.
Sono separati da virgole.

Cosa è il font stack?
Nel caso il primo carattere non fosse installato (Times New Roman), verrà utilizzato il secondo e così via creando quindi una sorta di stack. L'idea principale del font-stack è quella di utilizzare famiglie di caratteri simili, in maniera da consentire una visualizzazione di pagina simile per ogni sistema e/o browser.

Desktop first e mobile first?
Con il termine “Mobile First” ci si riferisce ad una strategia basata sul modificare il design della propria pagina web pensandola per la visualizzazione da dispositivi mobili.

Cosa è il collasso dei margini?
Vince chi ha il margine maggiore, non succede se è float o absolute.

Architettura di nodejs com’è fatta, quanti thread ha? Ha le code?
Ogni richiesta viene messa in una cosa e il server ha un unico thread che serve la coda. Si possono lanciare delle operazioni asincrone.

Differenza tra margin e padding
Margin (dall’inglese, margine) è una proprietà del CSS che definisce lo spazio tra un box e ciò che lo circonda all’interno della pagina web. Il padding indica, invece, lo spazio tra il contenuto e il box che lo contiene.

Border-box
Il valore border-box (al contrario del valore di default content-box) rende il bordo l'elemento ultimo del nostro box e fa sì che la larghezza consideri padding e bordi come interni.

Head e Header
HEAD continue i metadata e l’header è un segmento di pagina e si trova nel body.

CSS position: static
È il valore di default, quello predefinito per tutti gli elementi non posizionati secondo un altro metodo. Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.

CSS position: relative
L'elemento viene posizionato relativamente al suo box contenitore. In questo caso il box contenitore è rappresentato dal posto che l'elemento avrebbe occupato nel normale flusso del documento.

CSS position: absolute

L'elemento, o meglio, il box dell'elemento, viene rimosso dal flusso del documento ed è posizionato in base ai valori forniti con le proprietà top, left, bottom o right.
Il posizionamento assoluto (position: absolute;) avviene sempre rispetto al box contenitore dell'elemento.

Questo è rappresentato dal primo elemento antenato (ancestor) che abbia un posizionamento diverso da static.

CSS position: fixed
Usando questo valore, il box dell'elemento viene, come per absolute, sottratto al normale flusso del documento. La differenza sta nel fatto che per fixed il box contenitore è sempre la cosiddetta viewport. Con questo termine si intende la finestra principale del browser, ovvero l'area del contenuto. Altra differenza fondamentale: un box posizionato con fixed non scorre con il resto del documento. Rimane, appunto, fisso al suo posto.

Float
Muove un elemento tutto a destra o tutto a sinistra permettendo agli atri elementi di circondarlo.
- Si staccano dal flusso normale ma influenzano il contenuto dei blocchi intorno.
- Sono contenuti nell'area del contenuto dell'elemento che li contiene.
- I margini sono mantenuti.

Media query
Definisco degli stili per determinati media e device.

I siti Responsive forniscono layout diversi a seconda della dimensione della finestra di visualizzazione.
Un solo documento per tutti i device ma con stile variabile a seconda dello schermo.

Tecniche responsive
Elementi per un sito responsive:
1. Controllo viewport: Tag meta da inserire nell'head html
<meta name="viewport" content="width=device-width, initial-scale=1">
Opzioni
– width, height -> device-width o device-height oppure px
– user-scalable -> no, yes
– initial-scale -> 1
– minimum-scale -> 1 non scala
– maximum-scale -> 1 non scala
2. Controllo Layout con media queries: utilizzo i breakpoints. I breakpoint sono punti su una scala ideale di larghezza del viewport in cui si verifica una qualche modifica al layout della pagina. I breakpoint si definiscono con valori numerici nelle media query con max-width e min-width.
3. Media "fluidi"
img { max-width: 100%; } Controllo della larghezza:
– width: L'immagine si espande al 100%
– max-width: Se il container è più grande della dimensione in pixel dell'immagine si ferma il resize
– non posso impostare width ed height

FORM
HTTP GET avremo i dati concatenati nella URL – si usa ? all'inizio e & fra i dati.
Con POST i dati vengono inviati nel body della richiesta.

Chi è il this nel lato function?
Un oggetto può avere come proprietà una funzione. La parola chiave this usata dentro la funzione indica l'oggetto che la contiene, dipende dal contesto, la stessa funzione puo' indicare con this oggetti diversi

Cos'è una promise?
Una promise è un oggetto usato come contenitore per un risultato futuro di un'operazione asincrona risultato che puo' essere positivo o negativo a seconda se risolta

Come crearla?
Let promise = new promise (function(resolve,reject)) {}

Prototipo: In js gli oggetti hanno un prototipo che è un altro oggetto da cui ereditano tutte le proprietà

Come faccio a creare oggetti: normalmente con let jimmy = {prop: "valore"} oppure con una funzione che imposta le proprietà degli oggetti, questa funzione si chiama costruttore

Prototipo di funzione: è l'istanza di un oggetto che diventerà il prototipo per tutti gli oggetti creati usando il metodo costruttore

Prototipo di oggetti: è l'istanza dell'oggetto dal quale l'oggetto è ereditato

Come associare un prototipo ad un oggetto? Var tommy = new PIG ("Tommy", "pink") funzione costruttore

Scrivendo pig.prototype.color ="pink" assegniamo una proprietà a quell'oggetto
let user = new user("pippo") cambiano i valori del nuovo oggetto con i valori nuovi assegnati

Cosa è un metodo
I metodi rappresentano attività che un oggetto puo' compiere
Nel caso get e post che sono metodi http specificano la tipologia della richiesta

Il metodo costruttore è un metodo che serve a creare oggetti uguali o simili

Cosa è la coda di microtask?
Esempio lotteria

- La coda è primo-dentro-primo-fuori: i task messi in coda per primi sono eseguiti per primi.
- L'esecuzione di un task è iniziata solo quando nient'altro è in esecuzione.

Oppure, per dirla in modo semplice, quando una promise è pronta, i suoi gestori then/catch/finally sono messi nella coda. Non vengono ancora eseguiti. Il motore JavaScript prende un task dalla coda e lo esegue, quando diventa libero dal codice corrente.
La coda delle microtask ha priorità rispetto alle callback.

come si costruisce una rest api dal punto di vista handpoints

risorse utenti create update

update come la si fa

Struttura (HTML)
Presentazione (CSS)
Comportamento (JS)

Transpilers
Servono a tradurre linguaggi differenti, in una specifica versione di JS (source-to-source translators).

Polyfill
I polyfill sono librerie js che adattano del codice (html, css, js) per renderlo retrocompatibile con standard "vecchi".

Garbare collector
Un “garbage collector” rimuove le variabili che non ci servono dalla memoria automaticamente.

Scope
Lo scope è la visibilità di una variabile:
– La regione del nostro codice dove possiamo usare il nome della variabile/funzione
• Variabili locali definite dentro una funzione hanno lo *scope relative* al blocco della funzione stessa (local scope)
• Quando definiamo una variabile fuori da ogni funzione, è definita nel *global scope* e diventa visibile da ogni altro javascript che gira nella pagina

Oggetti
Un oggetto è una lista di coppie “proprietà” “valore”, racchiuse in parentesi angolari { }

Metodi
Un oggetto puo' avere tra le sue proprietà anche delle funzioni che chiameremo metodi.

This
Puo' essere utile nei metodi riferirci ad altre proprietà dell'oggetto.

Costruttori
Per creare oggetti uguali o simili possiamo usare delle funzioni.
Quando viene chiamato un costruttore con new:
• Viene creato un oggetto vuoto e assignato a this
• Viene eseguita la funzione
• Viene ritornato this

Lo scope di **var** è il functional block più vicino. Lo scope di **let** è l’enclosing block più vicino

Ripasso: costruttore
Posso creare un oggetto normalmente ...
let jimmy = {name: "Jimmy", color: "pink", age: 0};
let timmy = {name: "Timmy", color: "pink", age: 0};
Oppure posso usare una funzione che imposta le proprietà dell'oggetto
function Pig(name, color) {
this.name = name;
this.color = color; this.age = 0; }
var tommy = new Pig("Tommy", "pink");
Questa funzione si chiama "costruttore" – E' una funzione normalissima, ma la usiamo per "costruire" un oggetto

DOM
Fornisce una mappa strutturata del nostro documento ed i metodi per interfacciarsi con gli elementi.
• Ogni elemento della pagina è un nodo
• L’elemento radice è “document”
• “document” ha una serie di proprietà standard

Codice sincrono
Il codice è eseguito linea dopo linea

• Ogni linea aspetta che finisce l'esecuzione della precedente
• Le operazioni lunghe bloccano l'esecuzione del programma

Codice asincrono
• La funzione è eseguita alla fine dell'esecuzione del task
• Il codice sincrono continua la sua esecuzione
• Le immagini sono caricate in modo asincrono

Promises
Una Promise è un oggetto usato come placeholder per il risultato futuro di una operazione asincrona:
– Un contenitore per un valore assegnato in modo asincrono – Un contenitore per un valore futuro

Microtask
Per prima cosa,ogni volta che un task esce,il ciclo degli eventi controlla se il task sta restituendo il controllo ad altro codice JavaScript.In caso contrario,esegue tutti i microtask nella coda dei microtask.La coda dei microtask viene,quindi,processata più volte per ogni iterazione del ciclo degli eventi,anche dopo aver gestito eventi e altri callback.

In secondo luogo, se un microtask aggiunge più microtask alla coda chiamando queueMicrotask() , quei microtask appena aggiunti vengono eseguiti prima che venga eseguita l'attività successiva . Questo perché il ciclo degli eventi continuerà a chiamare i microtask finché non ne rimangono più in coda, anche se ne vengono aggiunti altri.
La ragione principale per usare i microtasks è questa:assicurare un ordine coerente dei compiti,anche quando i risultati o i dati sono disponibili in modo sincrono.

Fetch
– sono API basate sulle promise per le richieste ajax
– sostituiscono le XMLHttpRequest
– supportato da tutti i browser moderni

NODEJS
Può essere considerato come un ambiente runtime per JavaScript costruito sopra il motore V8 di Google.
• Ci fornisce un contesto dove possiamo scrivere codice JavaScript su qualsiasi piattaforma dove Node.js può essere installato
• L'ambiente ideale dove usare node.js è il server

Per Routing si intende determinare come un'applicazione risponde a una richiesta client a un endpoint particolare, il quale è un URI (o percorso) e un metodo di richiesta HTTP specifico (GET, POST e così via).

Architettura REST
REST (REpresentational State Transfer)
• insieme di linee guida o principi per la realizzazione di una architettura di sistema
– uno stile architetturale
• non si riferisce ad un sistema concreto
• non si tratta di uno standard

Principi REST
• Identificazione delle risorse

- Utilizzo esplicito dei metodi HTTP
- Risorse autodescrittive
- Collegamenti tra risorse
- Comunicazione senza stato

Come si fa un API rest

Con REST, vengono identificate delle risorse, ossia degli aggregati di informazioni che il servizio web offre. Se ad esempio parliamo di un e-commerce che vuole rendere pubblici i prodotti in vendita e i relativi prezzi potremmo individuare come risorse il singolo prodotto, una lista di prodotti selezionati, etc. Tali risorse vengono condivise in Rete mediante protocollo HTTP e proprio qui sta l'idea geniale alla base di REST: al suo interno HTTP possiede già tutto ciò di cui uno scambio di informazioni necessita pertanto non c'è bisogno di creare alcuna ulteriore sovrastruttura. Sfruttando gli elementi interni a questo protocollo potremo instaurare un dialogo tra il client ed il server.

Il protocollo HTTP funziona mediante un modello richiesta/risposta. Un client invia una richiesta al server, il server risponde. La richiesta può includere al suo interno una serie di funzionalità. Si può chiedere solo di leggere dati, di modificarli, crearli o cancellarli. Tutto ciò viene specificato mediante un campo della richiesta HTTP detto metodo. In REST si usano per lo più quattro metodi che indicheranno quali di queste operazioni il client starà richiedendo al server: con il metodo GET si chiederà solo di leggere dei dati, con il POST di crearne di nuovi, con il PUT di modificarli e con il DELETE di cancellarli.

API application programming interface