



Coding Bootcamp

Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi

Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi

Ci sono eccezioni che fanno la regola e poi ci stanno i programmatori che usano delle regole per sollevare eccezioni.

Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi

Try & catch / local storage

*Non importa quanto bravi o veloci siamo a scrivere codice.
Comunque vada, il nostro codice avrà sempre qualche errore...*

E indovinate un pò: possiamo gestire anche gli **errori**!

Javascript nella specificità del suo linguaggio utilizza le parole chiavi:

try e catch

Qui va il codice da testare,
se tutto ok, procede e la cosa finisce lì.

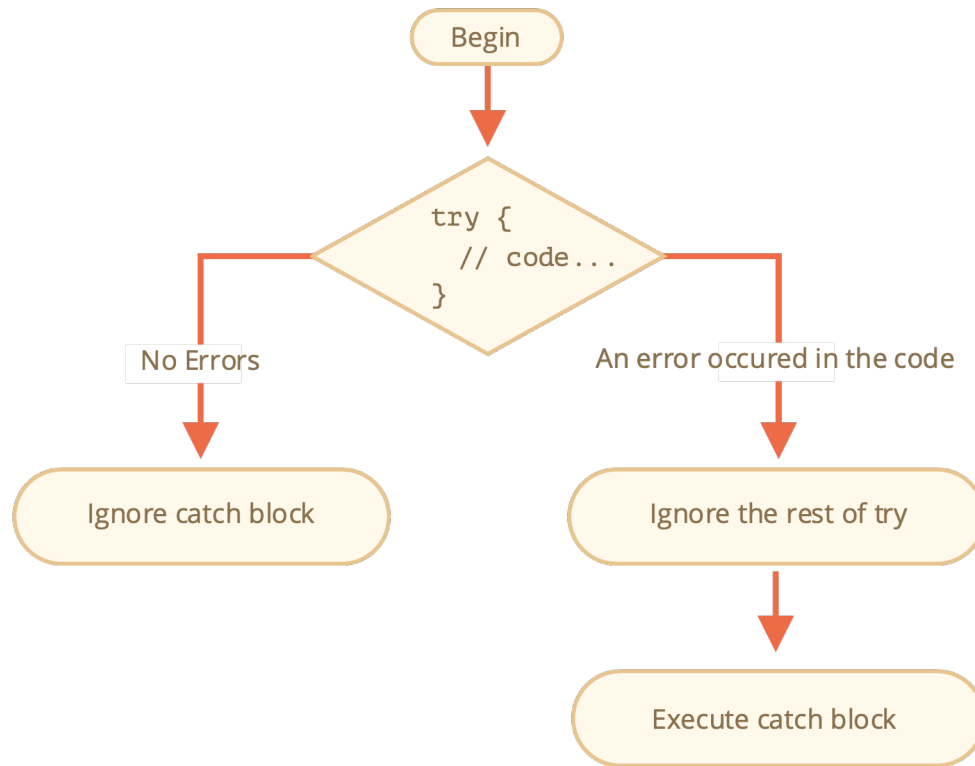
Se va male...

Passa allo statement **catch**



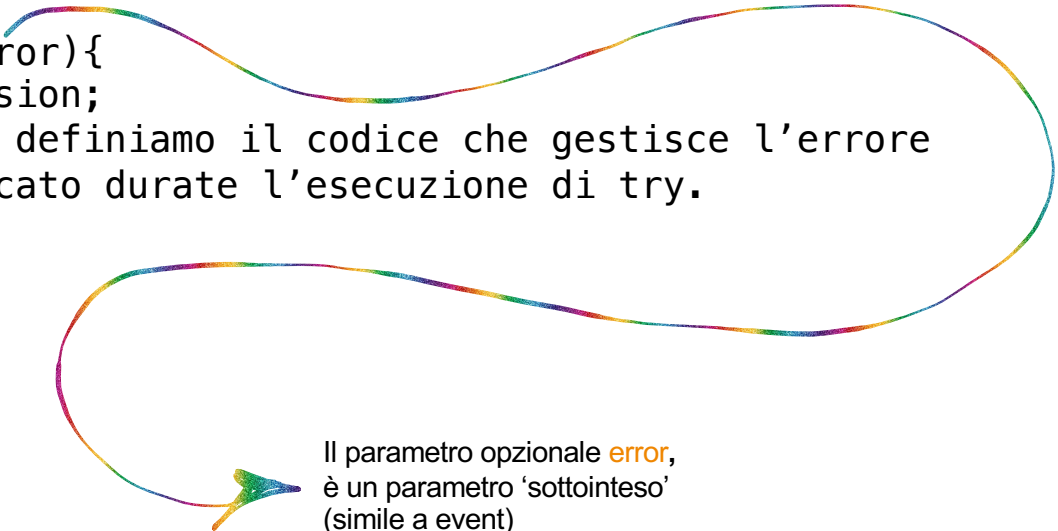
Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi



Try & catch, sintassi

```
Try {  
    expression;  
} //qui definiamo il codice (classico)  
  
catch(error){  
    expression;  
} // qui definiamo il codice che gestisce l'errore  
    // pescato durante l'esecuzione di try.
```



Il parametro opzionale **error**,
è un parametro 'sottointeso'
(simile a event)
Contiene l'errore rilevato.

Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi

Il local storage

Qualcuno aveva detto per caso: *peccato, perdiamo sempre tutti i dati al refresh della pagina!!*

Da oggi non più!

Il **localStorage** è una **API** già presente all'interno del vostro browser, che *ci fornisce una piccola locazione di memoria entro la quale archiviare ogni tipo di dato accettato dal linguaggio.*

Funziona in modo identico agli oggetti, dunque la sua struttura è identificata/formata da una coppia **chiave / valore**.



Quante opzioni ci restano...

Possiamo gestire il nostro **localStorage** con questi comandi:

- **setItem()** => aggiunge una chiave e un valore
- **getItem()** => cattura un dato definito
- **removeItem()** => elimina un dato definito
- **clear()** => pulisce l'intero localStorage
- **key()** => ritorna la chiave di un dato definito



setItem()

è il metodo che ci permette di immagazzinare dati. Questo **accetta due parametri**, una chiave e un valore:

```
window.localStorage.setItem('name', 'Edgemony');
```

getItem()

ci permette di pescare un element dal localStorage, specificando come **unico parametro** la chiave:

```
window.localStorage.getItem('name');
```


removeItem()

esattamente all'opposto di setItem(), ci permette di rimuovere un elemento definito all'interno dello storage.

Accetta **un solo parametro**, ovvero la chiave:

```
window.localStorage.removeItem('name');
```

clear()

simile a removeItem() ma dalla portata di una bomba atomica! Rimuove ogni singolo dato dallo storage.

Non accetta **alcun parametro**:

```
window.localStorage.clear();
```

I limiti del localStorage

Ogni cosa a questo mondo ha il proprio limite:

- **Mai immagazzinare dati sensibili** all'interno del local storage, verrebbero letti in chiaro da chiunque;
- **Non è assolutamente il sostituto del database** (server-side);
- **Limite di 5MB!** (circa);
- **Non ha alcun tipo di protezione** e può essere richiamato in ogni parte del codice;
- **È sincrono**, significa che ogni operazione verrà eseguita una dopo l'altra.



Try & catch / local storage

Immagazzinare dati lato client e 'salvaguardare' alcuni processi

