26° LEZIONE AULAB (15/01/25)

fondamenti come base per le ANIMAZIONI ... JS viene eseguito istantaneamente ... (quando la pagina viene caricata, con tutte le risorse...)

POSIZIONI = per JS = "a che punto è la vostra finestra del browser" / ... l'effettiva posizione all'interno del vostro foglio... se un elemento è visibile o no in uno specifico istante.. (posizione di un elemento del DOM)

(Ex): scroll = "andare su e giù" dimm applicativo (pagine) (INTRODUZIONE ALLE COORDINATE PERCHÉ OGNI SITO HA COORDINATE X/Y

(Ex): window.addEventListener("scroll", () => { console.log(window.scrollY); console.log(window.scrollX); }) // document = DOM NELLA SUA INTERFACCIA

HI STAMPO LE COORDINATE DELLA COSA CHE DESIDERO RISPETTO ALLA NOSTRA PAGINA, quindi, posso accedere a:

(Ex) innerHeight: altezza della finestra ... l'ALTEZZA INTERNA STANDARD

window.innerHeight "QUANTO L'UTENTE VEDE" ... rappresenta più l'altezza del viewport

viewport-height
viewport-width

(Ex) offset(ls) ≠ scroll

offset(ls): rappresenta la posizione SPECIFICA RISPETTO AL CONTENITORE GENITORE (specie in contesti più annidati)

scroll: si riferisce invece più alla PAGINA alla finestra (WINDOW) // "NELLO SPAZIO" (LA POSIZIONE) NON NELLA FINESTRA

(Ex) (html)

(JS)
```
let div = document.querySelector("div");
console.log(div.offsetHeight) // ALTEZZA DEL NODO -> mi dà 50px
console.log(div.offsetTop) // DISTANZA DAL TOP DELLA PAGINA
console.log(div.offsetLeft) // DISTANZA DAL BORDO SINISTRO
```

... possiamo usare più cose insieme: per capire se l'utente è in un determinato punto sto vedendo l'elemento o meno... (scroll + innerHeight)

in JS: "mi vedo è nella finestra dell'utente?"

```
let elementTop = div.offsetTop;
let elementBottom = elementTop + div.offsetHeight
let windowTop = window.scrollY;
let windowBottom = windowTop + window.innerHeight;
window.addEventListener("scroll", () => {
    (elementTop + div.offsetHeight) >= window.scrollY && div.offsetTop <= (...
    if (elementBottom >= windowTop && elementTop <= windowBottom
        console.log("Il div è visibile");
    }
    else {
        console.log("Non è visibile");
    }
})
```

(entrambi)

TECNICA PER VEDERE SE L'ELEMENTO È VISIBILE

Qui confronterà gli elementi e vede se la distanza di elemento Bottom dal basso non è maggiore della distanza della finestra...

PROVIAMO A GIOCARE CON I CONTATORI: quando c'è l'intersezione tra finestra e contenuto della pagina...

EX

```
<html>
  <div class = "container-section">
    <div class = "counter" data-target = "1500">0</div>
    <" ... " ... ... " = "3000"> " "
    <" ... " ... ... " = "4500"> " "
    <" ... " ... ... " = "6000"> " "
  </div>
  <script src = "./main.js" > ...
```

(CONTATORI CON I VALORI DI ARRIVO)

nell'head:

```
<style>
.counter-section {
    display: flex;
    justify-content: space-around
    padding: 50px;
    margin-top: 150px;
}
.counter {
    font-size: 3rem;
    font-weight: bold
}
```

JS

Voglio sia una cosa controllata (dentro l'event "load") CONTINUAMENTE

document.addEventListener("DomContentLoaded", () => {
CATURO I COUNTER
const counters = document.querySelectorAll(".counter");

FUNZIONE PER ANIMARLI (OGNI CONTATORE)
const animateCounter = (counter) => {
PARTO DA 0
let currentValue = 0;
PRENDO IL TARGET
const target = +(counter.getAttribute("data-target"));
+ = CONVERTE LA STRINGA IN INT
CALCOLO GLI INCREMENTI AD OGNI GIRO
const increment = Math.ceil(target / 100);
↳ anche 100000
};

FUNZIONE PER FARE L'UPDATE DEL COUNTER
const updateCounter = () => {
currentValue += increment;
if (currentValue >= target) {
currentValue = target;
counter.innerText = currentValue;
return;
};
FERMO L'ANIMAZIONE QUANDO RAGGIUNGE IL TARGET

counter.innerText = currentValue;

[if((currentValue) < target) {] = NON SERVE (anche inutile)
= //requestAnimationFrame
requestAnimationFrame(updateCounter);
};
la parte dei METODI RICORSIVI
(sincronizza l'animazione col refresh del browser (circa 60fps) garantendo animazioni fluide ed efficienti)

[requestAnimationFrame(updateCounter);] NON SERVE
updateCounter();
};
(chiamo la funzione ogni volta per continuare ad effettuare l'incremento)

Ora abbiamo fatto la nostra funzione per animare il nostro contatore ora dobbiamo far sì che parta solo quando c'è l'intersezione con la finestra...
FUNZIONE CHE CI RITORNA UN BOOLEANO (True/False) PERCHÉ VEDO IL NODO O MENO SULLO SCHERMO
const visibleElement = (node) => {
const elementTop = node.offsetTop;
const elementBottom = node.offsetTop + node.offsetHeight;
const windowTop = window.scrollY;
const windowBottom = windowTop + window.innerHeight;
return elementBottom >= windowTop && elementTop <= windowBottom
};
FUNZIONE CHE COMANDA TUTTO

PER OGNI COUNTER CONTROLLA SE LO VEDO (E NON E' GIA' ANIMATO) SE E' COSì FAI PARTIRE L'ANIMAZIONE

```
const handleScroll = () => {
    counters.forEach((counter) => {
      if(weSeeElement(counter) && !counter.classList.contains("animated")) {
          counter.classList.add("animated");
          animateCounter(counter);
      }
    });
};
window.addEventListener("scroll", handleScroll);
handleScroll();
```

sullo "scroll"

CONTROLLA GLI EVENTI / GLI ELEMENTI VISIBILI AL CARICAMENTO

DOMContentLoaded => evento del document = istante in cui il DOM viene caricato (carico solo l'albero del DOM)

load => evento della window istante in cui tutte le risorse vengono caricate (immagini, video...) (librerie CSS/JS)

Ex

```
document.addEventListener("DOMContentLoaded", () =>{
        let h1 = document.querySelector("h1");
        console.log(h1, "H1 dentro al DomLoaded");
window.addListener("load", () => {
        let h1 = document.createElement("h1");
        h1.innerText = "ciao Hackademy";
        document.querySelector("body").appendChild(h1);
})
```

... se non lo vediamo (h1) è perché questo viene creato dopo, ecco perché serve a contenuti fissi

i LOADER pronti per quando deve caricare le pagine (le "rotelline del caricamento")

infatti poi servirà una PROMISE ...

Se voglio elementi (loghi) che scorrono ...

(JS)
```
document.addEventListener("DOMContentLoaded", () => {
  const track = document.querySelector(".brand-scroll-track");
  const cloneItems = () => {
    let items = document.querySelectorAll(".brand");
    items.forEach((i) => {
      let clone = i.cloneNode(true);
      track.appendChild(clone);
    })
  }
  cloneItems();
  track.addEventListener("mouseenter", () => {
    track.style.animationPlayState = "paused";
  })
  track.addEventListener("mouseleave", () => {
    track.style.animationPlayState = "running";
  })
})
```

METODO INTERNO PER CLONARLI DIRETTAMENTE (ciclando un array di nodi ...)

→ se metto il puntatore sopra ai loghi si FERMA

↘ qui riparte quando tolgo il puntatore da sopra il logo

(html)
```
<div class="brand-scroll">
  <div class="brand-scroll-track">
    <div class="brand">
      <img src="....">
    </div>
  </div>
</div>
```

n-volte quanti sono i loghi

src = sorgente dell'immagine

(nel CSS)

```
img {
    width: 50px;
    height: 50px;
    object-fit: contain;
}
```

max-width ?
max-height ?

```
.brand-scroll {
    width: 100%;
    background: white;
    padding: 20px 0px;
    overflow: hidden;
    position: relative;
}
```

→ farà nascondere da un lato e poi le farà ricomparire (comparirano i CLONI)

```
.brand-scroll-track {
    display: flex;
    animation: scroll 15s linear infinite;
}
```

```
.brand {
    display: flex;
    gap: 15px;
    margin: 0 30px;
    align-items: center;
    justify-content: center;
}
```

→ flex-direction: column; ? (se serve)

ANIMAZIONE

```
@keyframes scroll {
    0% {
        transform: translateX(0);
    }
    100% {
        transform: translateX(-50);
    }
}
```