**EL3-8\_4ST TRIANGOLI**

**Autore: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Data: \_\_\_\_\_\_\_\_\_Classe: \_\_\_\_**

**ATTENZIONE**

Il significato dei simboli usati in questa e nelle prossime esercitazioni è spiegato dettagliatamente nella guida <http://www.classiperlo.altervista.org/Materiale/Generale/Simboli.doc> (scaricala e consultala in caso di dubbi).

|  |
| --- |
|  |

Il simbolo della manina in colore blu indica una domanda alla quale bisogna OBBLIGATORIAMENTE rispondere scrivendo in colore blu.

|  |
| --- |
|  |

Il simbolo della manina in colore blu con la scritta Cou New indica un codice che va incollato usando Courier New in colore blu.

|  |
| --- |
|  |

 Il simbolo della manina in nero indica una o più immagini o schermate da incollare (protette col tuo watermark, le tue iniziali di Nome e Cognome)

Il simbolo della manina con colori attenuati indica un'operazione che bisogna svolgere, senza rispondere a nessuna domanda (non vuol dire che non devi fare nulla - significa solo che non devi scrivere niente!).

Il simbolo della manina col ciak video indica un video da registrare con *Gif Recorder* (<http://gifrecorder.com/>)e da salvare in formato gif nella cartella dell'esercitazione.

recupero.

Questo simbolo indica un video di esempio o di spiegazioni da guardare su YouTube

**A) OPERAZIONI PRELIMINARI**

A1) Compila l'intestazione del documento scrivendo Nome, Cognome e Classe

A2) Crea una sottocartella di ES3 con nome uguale a quello di questa esercitazione (*EL3-8\_4ST Triangoli*)

A3) All'interno della sottocartella *EL3-8\_4ST Triangoli* salva questo file Word

**B) LA LIBRARY DEL JAVASCRIPT: FUNZIONI MATEMATICHE**

Le funzioni sono talmente importanti per il JS (come anche per gli altri linguaggi di programmazione) che alcune di esse sono state incorporate all'interno del linguaggio. In pratica è successo come sulle automobili: alcune cose che inizialmente erano degli optional (che si potevano aggiungere, a pagamento) sono poi diventate di serie su tutte le vetture (pensa per esempio all'air bag oppure agli alza cristalli elettrici).

Le funzioni incorporate si chiamano **funzioni di library** (in italiano sarebbe "funzioni di biblioteca", ma il termine viene spesso malamente tradotto con "funzioni di libreria").

|  |
| --- |
|  |

B1) Dove sta l'errore? Come si dice in inglese libreria?

Esistono funzioni di library per scopi diversi: per l'elaborazione dei testi, per la grafica, per il controllo del computer etc. In questa esercitazione esamineremo la categoria di funzioni di library più semplice e una delle più utili: le **funzioni matematiche**.

Eccoti qui i riferimenti per la spiegazioni e gli esperimenti su *Programmiamo* e *W3School*:

 <http://www.programmiamo.altervista.org/js/math/math1.html>

 <https://www.w3schools.com/js/js_math.asp>

Le funzioni matematiche sono quelle che vengono utilizzate per i calcoli matematici, come per esempio:

y = Math.sqrt(x);

Questa funzione calcola la radice quadrata del numero contenuto in x e mette il risultato nella variabile y. In notazione matematica

$$y=\sqrt{x}$$

Osserva subito una cosa particolare: in JS tutte le funzioni matematiche sono precedute dalla parola chiave Math seguita dal punto. Per esempio:

y = Math.pow(base, esp);  // eleva base a esponente

y = Math.round(x); // arrotonda x all'intero più vicino

Questo tipo di funzioni (precedute da qualcosa seguito dal punto) in JS si chiamano **metodi**. Per adesso non preoccupiamoci troppo di questo particolare. Per noi i metodi non sono altro che funzioni normali. L'unica differenza è che sono scritte in modo un po' diverso.

**FUNZIONI IN JAVASCRIPT E FUNZIONI IN MATEMATICA**

Avrai certamente notato che la parola **funzione** viene usata in matematica e nei linguaggi di programmazione in due modi simili ma leggermente diversi.

Per la matematica una funzione è una relazione fra due insiemi di numeri (dominio o campo di esistenza e codominio della funzione). La funzione matematica permette di associare ogni valore appartenente al dominio con uno e un solo valore del codominio. Per esempio, nel caso della radice quadrata $y=\sqrt{x}$ il dominio è rappresentato da tutti i numeri reali maggiori o uguali a zero (x≥0). Il codominio è costituito da tutti i possibili risultati del calcolo della radice quadrata (per ogni x appartenente al dominio della funzione).

Un modo più semplice per vedere una funzione è considerarla un dispositivo nella quale si inserisce un valore (la x) e che ne fornisce un altro (la y). Per esempio, sempre considerando la radice quadrata, se inserisco x = 9 ottengo $y=\sqrt{9}=3$.

Anche le funzioni in informatica sono dispositivi che, dati certi numeri (i parametri della funzione), ne forniscono in uscita degli altri (il valore di ritorno). La principale differenza rispetto alle funzioni matematiche è che i valori inseriti (i parametri) possono essere più di uno o talvolta anche nessuno (mentre nelle funzioni matematiche, la variabile è sempre solo una: la x). Inoltre, come sappiamo, in javascript esistono funzioni senza valore di ritorno, mentre in matematica tutte le funzioni calcolano un risultato (la y).

Ma basta con le chiacchiere, vediamo subito come funzionano questi metodi... pardon, funzioni di library.

B2) → *math1.html (HTML + CSS + JS, insieme)*

Pagina contenente una casella di testo e un pulsante.

Nella casella di testo si inserisce un valore (es. 3.14). Premendo il pulsante il programma dice (con un alert) se il valore inserito è intero oppure con la virgola. Usa il solito trucco this.value='' per cancellare la casella di testo quando ci si clicca sopra.

<https://youtu.be/YYH_lHQEwdo>

|  |
| --- |
|  |

B3) → Schermata browser

**COME SI FA**

Per effettuare il calcolo usa la funzione della library matematica Math.floor, la quale tronca un valore all'intero precedente (es. 4.7 diventa 4). Confrontando il valore troncato con quello inserito nella casella di testo, si può sapere se il numero è intero oppure no.

Per usare Math.floor non devi copiare e incollare il codice di nessuna funzione: le funzioni di library, come ti ho detto, sono incorporate nel linguaggio e perciò per usarle basta scrivere una chiamata. In pratica le usi come fossero delle normali istruzioni!

|  |
| --- |
|  |

B4) → Codice pagina

|  |
| --- |
|  |

B5) → *math1.gif* (Test completo del programma)

**C) IL TRIANGOLO RETTANGOLO**

Vediamo adesso un esempio un po' più complesso (e interessante) che usa le funzioni della library matematiche di JS

C1) → *math2.html, math2.css, math2.js*

Pagina con tre caselle di testo (per i due cateti e l'ipotenusa di un triangolo rettangolo), un pulsante per il calcolo e un pulsante PULISCI (del quale ci occupiamo più avanti).

<https://youtu.be/-PEib6dy-GE>

La pagina funziona in questo modo:

* se si inseriscono i valori dei due cateti, viene calcolata l'ipotenusa
* se si inserisce il valore dell'ipotenusa e di un cateto, viene calcolato l'altro cateto

Le formule matematiche da usare (Teorema di Pitagora) sono:







dove i è l'ipotenusa e c1 e c2 sono i due cateti.

**ATTENZIONE**

L'utente ha a disposizione tre caselle di testo, ma deve usarne solo due per l'inserimento dei dati (es. inserire i due cateti oppure uno dei due cateti e l'ipotenusa).

Il programma si accorge di quale delle tre caselle di testo è rimasta vuota e deve riempirla col risultato corrispondente. Assicurati di aver capito bene questo punto prima di proseguire.

|  |
| --- |
| **COME SI FA**Bisogna quindi scrivere, dentro la funzione di interfaccia associata al click sul pulsante, una serie di if.. else if che controllano le seguenti condizioni:// Nessuna oppure solo una casella contiene valoriif ((cateto1==0 && cateto2==0) || (cateto2==0 && ipotenusa==0) || (cateto1==0 && ipotenusa==0)) { alert("Devi inserire due valori!"); }// Tutte e tre le caselle sono pieneelse if (cateto1!=0 && cateto2!=0 && ipotenusa!=0) { alert("Devi inserire solo due valori!");  }// Bisogna calcolare l'ipotenusaelse if (ipotenusa == 0) { ris = (cateto1\*cateto1) + (cateto2\*cateto2);ipotenusa = Math.sqrt(ris);document.getElementById("ipo").value = ipotenusa; }// Bisogna calcolare il primo catetoelse if (cateto1 == 0) { ... }// Bisogna calcolare il secondo catetoelse { ... }La sequenza degli if... else if funziona per esclusione, nel senso che, se il primo if è vero, gli altri non vengono eseguiti; se il secondo else if è vero, tutti quelli seguenti non vengono eseguiti e così via. Osserva che la sequenza termina con un semplice else, perché, se tutte le condizioni precedenti sono false, l'unica che rimane è per forza quella giusta. |

C2) Osserva questa sequenza di if... else if

if (voto < 4)

 {

 alert("Insufficienza grave");

 }

else if (voto < 6)

 {

 alert("Insufficienza lieve");

 }

else if (voto < 7)

 {

 alert("Sufficiente");

 }

else

 {

 alert("Bravo");

 }

|  |
| --- |
|  |

C3) Perché se voto = 5 viene eseguito solo alert("Insufficienza lieve")e non anche alert("Sufficiente")? In fondo 5 è minore di 6, ma è anche minore di 7 dunque, secondo questo ragionamento, dovrebbero essere eseguiti entrambi gli alert. Spiegami!

**PER FAVORE, INDENTA!**

Quando scrivi gli if, ricordati sempre che le istruzioni che dipendono dalla condizione dell'if devono essere indentate, cioè spostate a destra usando il tasto TAB:

if (voto < 4)

 {

 alert("Insufficienza grave");

 }

else if (voto < 6)

 {

 alert("Insufficienza lieve");

 }

E, mi raccomando, per indentare usa e non spazio spazio spazio spazio....

**PER FAVORE, DICHIARA TUTTE LE VARIABILI!**

È vero che in JS si può omettere la dichiarazione var e utilizzare una variabile senza prima dichiararla. Questa però non è mai una buona pratica di programmazione, perché, per esempio se sbaglio a scrivere il nome di una variabile e scrivo *pipo* invece di *pippo*, il JS pensa di poter usare una nuova variabile di nome *pipo* (non dichiarata).

**PER FAVORE, CONVERTI SUBITO I VALORI NUMERICI ACQUISITI!**

Per esempio, ricordati di fare

cateto1 = document.getElementById("c1").value;

cateto1 = Number(document.getElementById("c1").value);

oppure anche

cateto1 = Number(document.getElementById("c1").value);

o, anche più semplicemente:

cateto1 = +document.getElementById("c1").value;

|  |
| --- |
|  |

C4) Incolla qui sotto il codice JS della tua funzione di interfaccia (controlla di aver indentato correttamente, di aver dichiarato tutte le variabili e di aver convertito i valori numerici come spiegato nei box precedenti):

Adesso introduciamo qualche miglioramento. Se osservi con attenzione il video <https://youtu.be/-PEib6dy-GE> vedrai che la casella che contiene il risultato del calcolo viene colorata in sfondo giallo.

|  |
| --- |
|  |

C5) Introduci il miglioramento precedente (colore di sfondo giallo per la casella con il risultato) in tutti i casi (calcolo ipotenusa, calcolo cateto1, calcolo cateto2). Incolla qui sotto solo il codice per colorare di giallo la casella con l'ipotenusa:

Osserva anche nel video che c'è un pulsante PULISCI che ripulisce i valori contenuti in tutte le caselle di testo e rimette lo sfondo bianco dappertutto.

**COME SI FA**

Per ripulire i valori contenuti in tutte le caselle possiamo usare un trucco che abbiamo già visto. Se richiudiamo tutte le caselle in un elemento form, la pulizia può essere fatta inserendo un elemento input di tipo reset:

<form>

 <p>Cateto 1:<input type="text" id="c1"> </p>

 ...

 <input type="button value="CALCOLA">

 <input type="reset" value="PULISCI">

</form>

Il metodo precedente non pulisce però il colore di sfondo giallo (cancella solo i valori). Possiamo ottenere anche questo usando una funzione di interfaccia collegala col click sul pulsante reset e dentro alla funzione pulire il colore di sfondo:

document.getElementById("c1").style.backgroundColor = "white";

|  |
| --- |
|  |

C6) Introduci il miglioramento precedente e incolla qui sotto il codice completo della tua funzione di interfaccia collegata all'evento onclick sul pulsante PULISCI:

C7) → *math2.gif (test funzionamento completo)*

**D) DISEGNIAMO IL TRIANGOLO RETTANGOLO**

Vogliamo ora rendere più interessante il nostro programma, facendogli disegnare il triangolo rettangolo.

<https://youtu.be/aMVe8er8Xxk>

D1) →*math3.html, math3.css, math3.js* (collegate fra loro)

Vogliamo adesso mettere nella pagina due quadrati e delle stesse dimensioni. Uno dei due quadrati è realizzato con un div oppure con un form (in nero in figura) e contiene le caselle di testo e il pulsante; l'altro quadrato (in rosso in figura) è realizzato con un canvas.



**SUGGERIMENTI**

Puoi posizionare i due quadrati usando il posizionamento absolute o relativo (in questo caso ti consiglio di usare gli Strumenti per sviluppatori di Chrome per posizionare con precisione) oppure puoi usare float con left e right.

Ricordati di usare gli attributi e non gli stili per dimensionare il canvas.

|  |
| --- |
|  |

D2) → Schermata browser

D3) Come puoi vedere dal video, il programma calcola il valore dell'ipotenusa a partire dai cateti e lo visualizza in una casella readonly con sfondo colorato. Inoltre nella stessa casella viene visualizzato un messaggio di errore se i cateti non sono forniti oppure hanno valore negativo.

|  |
| --- |
|  |

D4) Scrivi la tua funzione di interfaccia collegata col pulsante CALCOLA nella quale, per adesso, ti limiti a calcolare l'ipotenusa e a visualizzarla nella casella colorata. Inoltre visualizza un messaggio di errore e non fare il calcolo, se i valori dei cateti non sono validi. Incolla il codice:

**SUGGERIMENTI**

Per verificare se i cateti sono validi basta controllare

if (cateto1<=0 || cateto2<=0) // se cateto1<=0 oppure cateto2<= 0

ATTENZIONE però: il controllo precedente funziona solo se ti sei ricordato di convertire correttamente i cateti in formato numerico!

Per interrompere la funzione senza svolgere i calcoli usa l'istruzione return.

A questo punto vogliamo fare in modo che, oltre a fare i calcoli dei lati come nell'esercizio precedente, il nostro programma disegni anche il triangolo:



A tale scopo puoi usare questa funzione:

// Disegna un segmento

// a partire dal punto di coordinate \_x1,\_y1

// fino al punto di coordinate \_x2,\_y2

// nel CANVAS di identificatore \_ident

function segmento(\_x1, \_y1, \_x2, \_y2, \_ident) {

 let c = document.getElementById(\_ident);

 let ctx = c.getContext("2d");

 ctx.beginPath();

 ctx.moveTo(\_x1, \_y1);

 ctx.lineTo(\_x2, \_y2);

 ctx.lineWidth = 2;

 ctx.stroke();

}

**COME SI FA**

Supponi di voler posizionare il vertice del triangolo nel punto di coordinate x=10 e y=10. Le coordinate dei tre segmenti da disegnare sono mostrate in figura:



In pratica devi usare la funzione segmento per tre volte, per disegnare i tre segmenti. Per esempio il cateto orizzontale si disegna così:

segmento(10,10,10+cateto1,10,"myCanvas");

dove cateto1 è la variabile che contiene la lunghezza in px del cateto e myCanvas è l'id che ho assegnato al canvas.

Inoltre, come puoi vedere dal video, il triangolo precedente viene cancellato ogni volta dal canvas. Per ottenere questo puoi usare la seguente funzione:

function cancella(\_ident)

{

 let c = document.getElementById(\_ident);

 let ctx = c.getContext('2d');

 ctx.clearRect(0, 0, c.width, c.height);

}

**PER FAVORE, NON CONFONDERE I NOMI DEI PARAMETRI CON I VALORI**

La famosa LEGGE DI PERLO N.3 dice che

i parametri delle funzioni di elaborazione sono semplici nomi segna posto e NON devono essere usati nel programma

Per esempio ident è un parametro, non è il nome che devi assegnare al tuo id! Se vedo un canvas dichiarato in questo modo, mi arrabbio:

<canvas id="\_ident" height="450" width="450"></canvas>

|  |
| --- |
|  |

D5) → Codice JS della tua funzione di interfaccia completa di tutto:

D6) → *math3.gif (test funzionamento completo)*

Un altro modo per disegnare il triangolo rettangolo consiste nell’usare l’elemento SVG e la grafica vettoriale.

D7) Copia *math3.html, math3.css, math3.js in math4.html, math4.css, math4.js* (e aggiorna i collegamenti).

D8) Sostituisci al posto di canvas l’elemento svg.

**DISEGNARE UN TRIANGOLO CON SVG**

Per disegnare un triangolo in grafica vettoriale puoi usare l’elemento polygon (che serve, in generale, per disegnare poligoni con un numero di lati qualsiasi). Per esempio:

<polygon points="200,10 250,190 160,210" fill="none" stroke="black" />

disegna un triangolo con vertici di coordinate (200,10) (250,190) (160,210).

 Le coordinate sono indicate con tre coppie di valori x,y con uno spazio di separazione fra una coordinata e l’altra. Gli attributi fill e stroke servono per specificare il colore di riempimento e quello del bordo (per un esempio più dettagliato vedi qui <https://www.w3schools.com/graphics/svg_polygon.asp> )

D9) Inserisci dentro l’elemento svg un triangolo nascosto (display:none) con coordinate e colori a piacere.

 <svg id="myCanvas" height="450" width="450">

 <polygon fill="none" stroke="red" style="display: none" id="triangolo"/>

 </svg>

Come vedi non ho assegnato alcun valore all'attributo points che serve per specificare i vertici del poligono. Infatti esso è nascosto (invisibile).

D10) Cliccando sul pulsante, dentro alla tua funzione di interfaccia bisogna:

1. rendere visibile l'elemento polygon che è stato precedentemente nascosto
2. assegnare i valori corretti ai vertici per disegnare il triangolo
3. in questo caso non serve la funzione cancella() poiché l'elemento svg viene cancellato automaticamente ogni volta

**COME SI FA**

Le coordinate dei vertici del poligono sono gli stessi valori che hai usato prima per disegnare i segmenti col canvas.

Supponiamo che le coordinate siano memorizzate in x1,y1 x2,y2 x3,y3:

document.getElementById("triangolo").setAttribute("points", x1+","+y1+" "+x2+","+y2+" "+x3+","+y3);

L'operatore + serve come sempre per concatenare. Attenzione agli spazi (blank) indicati con " " che sono fondamentali per separare le coordinate.

|  |
| --- |
|  |

D11) → *math4.js*

D12) → *math4.gif*

.

**E) GRAFICI FUNZIONI**

In matematica è spesso utile tracciare il grafico di una funzione (funzione nel senso matematico della parola), mettendo i valori della x sull'asse orizzontale e quelli della y sull'asse verticale.

Possiamo scrivere un semplice programma per il tracciamento della funzione seno in JS usando l'elemento canvas e le funzioni matematiche.

Anzitutto ecco le funzioni (nel senso informatico della parola) che ci servono (vedi pagina seguente):

let Ampiezza, Frequenza;

let colore = ["rgb(255,0,0)","rgb(0,255,0)","rgb(0,0,255)","rgb(0,0,0)"];

indcol = 0;

function fun(\_x) {return Ampiezza\*Math.sin(Frequenza\*\_x); }

function draw(\_ident) {

 var canvas = document.getElementById(\_ident);

 if (null==canvas || !canvas.getContext) return;

 let axes={}, ctx=canvas.getContext("2d");

 axes.x0 = .5 + .5\*canvas.width; // x0 pixels from left to x=0

 axes.y0 = .5 + .5\*canvas.height; // y0 pixels from top to y=0

 axes.scale = canvas.height/4; // 40 pixels from x=0 to x=1

 axes.doNegativeX = true;

 funGraph(ctx,axes,fun,colore[indcol],2);

 indcol++;

 if (indcol==4)

 indcol = 0;

}

function funGraph (\_ctx,\_axes,\_func,\_color,\_thick) {

 let xx, yy, dx=4, x0=\_axes.x0, y0=\_axes.y0, scale=\_axes.scale;

 let iMax = Math.round((\_ctx.canvas.width-x0)/dx);

 let iMin = \_axes.doNegativeX ? Math.round(-x0/dx) : 0;

 \_ctx.beginPath();

 \_ctx.lineWidth = \_thick;

 \_ctx.strokeStyle = \_color;

 for (let i=iMin;i<=iMax;i++) {

 xx = dx\*i; yy = scale\*\_func(xx/scale);

 if (i==iMin) \_ctx.moveTo(x0+xx,y0-yy);

 else \_ctx.lineTo(x0+xx,y0-yy);

 }

 \_ctx.stroke();

}

function showAxes(\_ident) {

 let canvas = document.getElementById(\_ident);

 if (null==canvas || !canvas.getContext) return;

 var axes={}, ctx=canvas.getContext("2d");

 axes.x0 = .5 + .5\*canvas.width; // x0 pixels from left to x=0

 axes.y0 = .5 + .5\*canvas.height; // y0 pixels from top to y=0

 axes.scale = canvas.height/4; // 40 pixels from x=0 to x=1

 axes.doNegativeX = true;

 let x0=axes.x0, w=ctx.canvas.width;

 let y0=axes.y0, h=ctx.canvas.height;

 let xmin = axes.doNegativeX ? 0 : x0;

 ctx.beginPath();

 ctx.strokeStyle = "rgb(128,128,128)";

 ctx.moveTo(xmin,y0); ctx.lineTo(w,y0); // X axis

 ctx.moveTo(x0,0); ctx.lineTo(x0,h); // Y axis

 ctx.stroke();

}

È probabile che tutto questo ti sembrerà molto complicato. Non temere. Non devi cercare di capire come funziona, devi solo sapere come lo si usa...

E1) → *math5.html, math5.css, math5.js*

<https://youtu.be/f7wgQVcqFDY>

La pagina deve contenere un CANVAS (con dimensioni a piacere), due caselle di testo e due pulsanti, come mostrato nel video.

E2) Per prima cosa tracciamo gli assi x e y con la funzione *showAxes*. Per poter usare questa funzione, devi semplicemente passarle l'identificatore del tuo CANVAS (fra virgolette). Non c'è nessun valore di ritorno:

showAxes("myCanvas");

Questa funzione deve però essere eseguita subito, al momento del caricamento della pagina e non può dunque essere collegata a un pulsante.

**ESEGUIRE UNA FUNZIONE AL CARICAMENTO DELLA PAGINA**

Esistono due modi diversi per eseguire una funzione al caricamento della pagina. Il primo metodo fa uso dell'evento onLoad solitamente associato al body:

<body onload="funzione()">

L'evento onload viene eseguito quando l'elemento corrispondente è caricato sul browser. Se, come detto, lo associ al body, il caricamento del body coincide col caricamento della pagina e dunque la funzione viene chiamata al caricamento della pagina.

Il secondo metodo consiste nello scrivere la chiamata alla funzione direttamente nel JS fuori da tutte le funzioni di interfaccia (nella stessa "zona" dove scrivi le variabili gloabali):

<script>

funzione(); // questa funzione viene eseguita al caricamento

let pippo, pluto: // variabili globali

function funzione()

{

}

...

</script>

Osserva che, usando questo secondo metodo, funzione() viene scritta due volte: la prima come chiamata - funzione(); - e la seconda per scrivere il codice della funzione stessa.

E3) Usando il metodo che preferisci, traccia gli assi cartesiani con la funzione *showAxes*

|  |
| --- |
|  |

E4) → Schermata browser con gli assi tracciati.

Occupiamoci ora del grafico. Le **variabili** *Ampiezza* e *Frequenza* servono per contenere i valori di ampiezza e frequenza della sinusoide, letti dalle due corrispondenti caselle di testo.

Nella tua funzione di interfaccia collegata al pulsante GRAFICO dovrai dunque inserire qualcosa del genere:

 Ampiezza = document.getElementById("ampiezza").value;

 Frequenza = document.getElementById("frequenza").value;

E5) La funzione per fare il grafico si chiama *draw* e basta passarle l'id del CANVAS (nessun valore di ritorno). La funzione *draw* dev'essere chiamata sul click sul pulsante GRAFICO.

|  |
| --- |
|  |

E6) → Codice della tua funzione di interfaccia

E7) Occupiamoci infine del pulsante PULISCI per cancellare tutto il canvas. Puoi riutilizzare la funzione cancella vista al punto D.

ATTENZIONE però che devi modificare leggermente la funzione cancella in modo tale che, dopo aver cancellato il grafico, ridisegni gli assi cartesiani.

E8) → *math5.gif*

Prova inserendo in sequenza: Ampiezza 1, Frequenza 1 - Ampiezza 1, Frequenza 2 - Ampiezza 1.5 Frequenza 3 (tre grafici sovrapposti - colori diversi). Prova anche il pulsante PULISCI.

**F) OPERAZIONI FINALI**

F1) Controlla di aver risposto a tutte le domande e incollato tutte le schermate. Tutte le caselline dovrebbero avere un segno X, per indicare che hai risposto 

F2) Comprimi le immagini contenute in questo file Word (seleziona un'immagine, scheda *Formato* e poi *Comprimi immagini* e infine *Applica a tutte le immagini del documento*) in modo da ridurne le dimensioni.

F3) Controlla che la cartella di questa esercitazione contenga i seguenti file con i nomi qui indicati:



|  |  |  |
| --- | --- | --- |
| **Nome del file** | **Tipo del file** | **Descrizione** |
| *EL3-8\_4ST Triangoli* | Word | Il file di questa esercitazione |
| *math1.html* | HTML+CSS+JS |  |
| *math1.gif* | GIF | Test funzionamento |
| *math2.html* | HTML |  |
| *math2.css* | CSS |  |
| *math2.jss (cambia estensione)* | JS |  |
| *math2.gif* | GIF | Test funzionamento |
| *math3.html* | HTML |  |
| *math3.css* | CSS |  |
| *math3.jss (cambia estensione)* | JS |  |
| *math3.gif* | GIF |  |
| *math4.html* | HTML |  |
| *math4.css* | CSS |  |
| *math4.jss (cambia estensione)* | JS |  |
| *math4.gif* | GIF |  |
| *math5.html* | HTML |  |
| *math5.css* | CSS |  |
| *math5.jss (cambia estensione)* | JS |  |
| *math5.gif* | GIF |  |

F4) Chiudi tutti i file, zippa la cartella di questa esercitazione e inviala all'insegnante su Classiperlo.