**JAVASCRIPT**

Cos’è? Come si usa? Dove si usa? (browser)

Sintassi, variabili, if, cicli, funzioni.

Oggetti, array, funzioni in dettaglio, elementi HTML

Come usare java nel browser?

Document Object Model, rappresentazione della pagine nella memoria del browser, e interagire con l’utente. Eventi → AJAX

Compatibilità tra i browser

Utilizzare gli oggetti è più semplice rispetto a C++. Ma sono più complicati i programmi..

Cos’è? È un linguaggio di programmazione, dinamico, interpretato e scriptato.

Linguaggio scripting: Piattaforma che supporta javascript, e usarlo per fargli compiere delle azioni. Le funzionalità sono numerose, e scrivendo semplici righe, è possibile creare qualcosa di enorme.

-------------------------------------------------------------------------------------------------------------

Interpretato perché di da il programma in pasto al browser, il quale lo esegue per conto suo, ma ultimamente è diventato compilato.

Linguaggio Clientside per pagine web dinamiche

Transpilare: compilare in un altro linguaggio, non linguaggio macchina, per fare si che supporti anche i browser più vecchi.

Di solito javascript non si compila. Ma si transpila. Scrivere programma con versione nuovissima, e transpilarlo nelle versioni vecchie.

Dinamicamente tipizzato (dynamically typed): il tipo delle variabili può cambiare, perché dipende dal valore che ci abbiamo messo dentro. Non si dichiara il tipo delle variabili, come PYTHON

(INT, CHAR, OGGETTO, STRING; FLOAT; BOOL)

NUMBER = tipo che racchiude tutti i tipi numerici

Interazione a eventi con utente e non solo all’interno del browser.

Un pezzo di codice parte quando si scatena un evento. Agisce in risposta agli eventi.

Il browser interagisce con l’utente. Input Output Serverside a eventi con NODE.JS

SINTASSI SIMILE A C++ e JAVA

**Garbage** **collection**: quando si creano degli oggetti dinamici con **.new()** non è necessario cancellarli con **.delete()**

Non si usano le classi ma un altro tipo di prog a oggetti.

Classe è una ricetta per N oggetti. C++

1 Oggetto che è il prototipo di tutti gli altri dello stesso tipo. Ciò è comodo perché permette di creare trucchetti. Si creano gli oggetti uguali, con le stesse caratteristiche.

**First-class function**: le funzioni possono essere interpretate come valori, perciò potranno essere salvate in variabili, funzioni di funzioni, f ( g(x) )

Il cambiamento di un prototipo influenza il figlio.

**STORIA DI JAVASCRIPT**

Compatibilità di JS. È standard. Ma se viene aperto da diversi browser?

Bisogna rendere tutto compatibile.

Nasce nel 1995.

1996

Netscape 2.0 (primo browser) con motore SpiderMonkey (uno dei primi in java)

Anche Internet Explorer 3 in JSCRIPT

1997

Standardizzazione di JAVASCRIPT, ECMAScritp (ECMA-262), tutti possono scrivere linguaggi che supportano questo ECMA

1998

ECMAScritp 2 Netscape 3

1999

ES3 e Netscape 4

2000

Microsoft usa come browser di default Internet explorer 5.0. Da qui deriva AJAX

XMLHTTP, base di AJAX

2004 Standard ES4,

Nascita di Firefox e Opera fonda il WHATWG, gruppo di lavoro che porta avanti gli standard di JS e HTML per migliorare

2007 ES4 fallisce perché Yahoo non lo supporta più, e tutti gli altri anche

ES3.1 in parallelo

2008 Google rilascia Chrome, nuovo browser.

For (**var** i=0; i<10; i++) {

console.log ( “number “ +i);

} JSCRIPT

si dichiarano le variabili con var (dinamicamente tipizzato)

For (**int** i=0; i<10; i++) {

system.**out**.println ( “number “ +i);

} JAVA

NOMI DELLE VARIABILI

document: contiene la struttura dati del doc corrente

getElementById = nome di una funzione da parte di document

$ è un carattere che si può usare per nominare variabili/funzioni/classe

NON è come PHP, che identifica le variabili

Un nome non può cominciare con un numero

NOME CASE SENSITIVE

PAROLE CHIAVE:

await, break, case, catch, class, const, continue, debugger, default, delete, do, else, enum, export, extends, false, finally, for, function, if, implements, import, in, int, instanceof, interface, let, new, null, package, private, protected, return, super, switch, this, throw, true, try, typeof, undefined, var, void, while, with, yield

class persona {

name= “jim”

age=23 }

vett [1,2,3] OPPURE [“ciao”, “come”, “va”] **ARRAY**

undefined, torna al concetto dinamicamente tipizzato. Valore vuoto, perché non ha ancora un valore.

NULL= valore vuoto, ma deciso.

{ nome: ”nome”, age: 23} **OGGETTO**

**PUNTEGGIATURA**: si usano molto, famiglia algof

/\* COMMENTO \*/

SCRICT MODE

“use strict” ABILITA NEL BROWSER UN ALTRO METODO

/\* CODICE \*/

Alcuni pezzi di codice vengono interpretati in maniera diversa.

Un errore fa crashare il mio script, perché così si capisce a quale riga vi è l’errore. Quando crasha dice la riga a cui c’è l’errore, e che genere di errore è

HTML è un linguaggio di MARKUP, non permette di andare a cambiare il flusso di esecuzione del programma.

Serve solo a definire la struttura di una pagina.

HTML + CSS + JAVASCRIPT = FRONT END

PROGRAMMAZIONE CARTELLA C:/JAVA/WORKSPACE

Bisogna dare ini pasto al browser un programma HTML per poi usare JS, che appunto si appoggia su HTML

Digitare html:5. Compare un file html già sviluppato

Tasto DX, Open with Live Server. Apre la pagina HTML nel browser, a IP:PORTA

(127.0.0.1 **:** 5500)

F12 nel browser per vedere codice sorgente pagina.

È possibile scrivere un pezzo di codice in un altro linguaggio all’interno di una pagina HTML

<script> CODICE </script> possono essere messia sia dentro <head> che dentro <body>

Dentro <head> però gli script devono essere più leggeri

CONSOLE.LOG(“TESTO STAMPATO NELLA CONSOLE”)

È uno strumento che è possibile usare come debug

ALERT(“ATTENZIONE”), invia un messaggio all’utente come notifica dicendo che c’è un’allerta.

Interagisce con l’utente.

SCRIVERE DENTRO LA PAGINA

Accedere al DOM (Document Object Model)

DICHIARAZIONE VARIABILI

Var nome -> variabile function scope

Let y=4, z=5 -> let funzione più o meno come var, solo che è block scope, cioè esiste solamente all’interno di un blocco, come su C++

Const x=3

Let y=4, z=5

{

Let x=2 // si può fare perche x avrà il valore 2 solo all’interno del blocco corrente

Y=8 // cambia la variabile y precedente al blocco

}

Console.log(x) // stamperà il valore 3 NON 2

DICHIARAZIONE FUNZIONI

Function nome() {

//ISTRUZIONI

}

BREAKING STATEMENTS

BREAK=lascia il blocco corrente e esce, fino al blocco più esterno (CICLI E SWITCH)

RETURN=lascia la funzione corrente e ritorna il valore indicato

CONTINUE: esce dalla iterazione corrente, all’interno del ciclo, cioè esegue quella successiva

THROW=fa crashare lo script, quindi si esce dallo scrip. Throw ritorna un valore

GESTIONE DEGLI ERRORI

ALERT (“NOTIFICA ALL’UTENTE”)

Throw new ERROR(“SI E VERIFICATO UN ERRORE”)

Try {

Istruzione

} catch (errore) { //errore è una variabile che conterrà l’errore che si verifica

Istruzioni in caso di errore

alert(errore.message)

console.error(errore) // stampa il vero e proprio errore

}

OGGETTI

Const oggetto = {

Nome: ”Damiano” //PROPRIETà o ATTRIBUTO

Cognome: ”Fumagalli”

} ;

console.log(oggetto) -> stampa l’intero oggetto con i suoi attributi

console.log(oggetto.nome) -> stampa l’attributo nome

è possibile aggiungere gli attributi da fuori del corpo dell’oggetto.

Oggetto.data = 32 -> crea attributo data con valore 32

Prompt(“DOMANDA”) -> effettua una domanda via notifica, e è possibile anche inserire un valore da tastiera, INPUT

Dato = prompt(“Inserisci l’attributo da visualizzare”)

Console.log ( oggetto[dato] )

oggetto[dato] -> cerca nell’oggetto l’attributo dato=”input da tastiera”

if ( “attributo” in oggetto) {

se l’oggetto possiede l’attributo “attributo” allora esegue il codice all’interno

}

CICLO CON OGGETTI

For (let i in oggetto) {

Cosole.log(i) -> stampa i ogni volta che la condizione è verificata, cioè se i è in oggetto

} -> i otterrà i nomi delle proprieta/attributi

Facendo console.log( i , oggetto[i]) stamperà il nome e il contenuto dell’attributo, DENTRO FOR

ARRAY

Sono un tipo speciale di oggetto

Const array = []; -> array vuoto

Const array = [1,2,3,4]

Length = lunghezza delle celle del vettore

Per svuotare un array è semplicemente necessario impostare length = 0

Const vett [

“matita”, “occhiali”, SEPARARE GLI ELEMENTI CON LA VIRGOLA

{ OGGETTO

Nome: “cane”,

eta: 32

}

];

Dentro un vettore è possibile annidare un OGGETTO, che a sua volta può avere un array

Console.log( vett [0] )

Se l’elemento dell’array è un oggetto, è possibile anche accedere agli attributi dell’oggetto

Array[0] . attr array[0] = oggetto oggetto.attr

EREDITà PROTOTIPALE: è possibile implementare oggetti già esistenti con altre funzioni/metodi

Le funzioni possono essere passato come argomenti in altre funzioni

Una Funzione può essere passata come parametro, se:

* nomeFun -> viene salvato il contenuto della funzione, quindi il codice
* nomeFun() -> viene eseguita la funzione e viene ritornato il valore

**3/3/2021**

DOM = struttura dati = Document Object Model

Developer.mozilla.org javascript reference

**JAVASCRIPT API** : funzionalità con interazione con l’utente

ECMA SCRIPT = standard del javascript

**WEB API:** client-side, nei browser

**EVENTI**

Oggetto EVENT = si crea quando si verifica un evento

LISTENER = qualcuno che ascolta e attende che si verifica questo evento

LISTENER + TARGET ( elemento sullla pagina )

CHE TIPO DI EVENTO CI ASPETTIAMO?

CALLBACK: pezzo di codice che vogliamo venga eseguito all’avvenimento dell’evento

Sarà una funzione, che verrà passata come valore, non eseguita.

Function logEVENT (evt) {

Console.log( evt.type , evt.taget, …)

};

Evt è un oggetto che rappresenterà l’evento in questione

<button id=”logEVENT”, onclick=”logEVENT(event);”>

INVIO

</button>

const button = document.createElement(“button”)

button.onclick = logEVENT; La funzione verrà passata come valore all’interno del pulsante