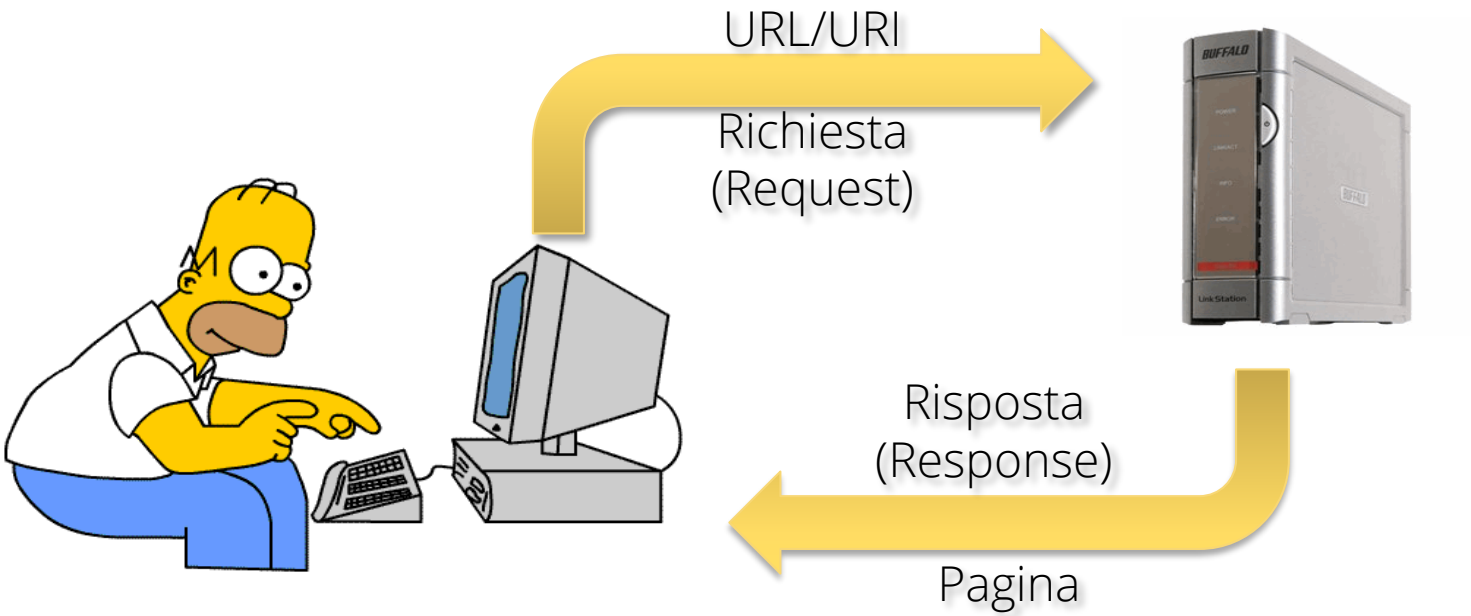
LEZIONE 5

Javascript

Programmare per il web significa scrivere almeno due applicazioni: una chiamata client e una chiamata server le quali, comunicando tra loro, producono un'applicazione web. Bisogna quindi imparare a programmare sia un'applicazione client sia un'applicazione server.
Questo libro si occupa di insegnare le tecniche necessarie alla realizzazione di un'applicazione client.

La programmazione web lato client ha il suo cardine nel web Browser!
Il Browser è infatti una potentissima applicazione client che noi possiamo usare per interagire con l'utente.
Tuttavia scrivere codice per il Browser comporta la conoscenza di molteplici linguaggi, ognuno con determinate caratteristiche. Il libro fornisce una visione di insieme di tali linguaggi soffermandosi in modo dettagliato su quelli che sono assolutamente necessari: HTML, CSS e JavaScript.

ARCHITETTURA CLIENT SERVER



ARCHITETTURA CLIENT-SERVER

- Server
  - Programma *in ascolto* su una porta (punto di accesso)
  - Quando arriva una richiesta da un client, il server analizza questa richiesta (eventualmente con l'aiuto di altri programmi), elabora una risposta (anche in questo caso, eventualmente con l'aiuto di altri programmi) e la invia al client.
  - Un server, generalmente, serve più client contemporaneamente
- Client
  - Un client è un programma che si connette ad un server, fa una richiesta ed aspetta una risposta. Quando la risposta arriva la elabora.

WEB SERVER

- Un **Web Server (server che fornisce** servizi sul Web) è sostanzialmente un **HTTP Server (server che comunica** mediante il protocollo HTTP) e **gestisce 2 flussi di informazioni:**
  - **le richieste in arrivo dai client** (HTTP request)
  - **le risposte del server, inviate** ai client (HTTP response)

BROWSER

- Un Web browser è un **HTTP client, cioè un** programma, dotato di interfaccia grafica, che:
  - **interagisce con un HTTP server , richiedendone** i servizi (per es. pagine Web)
  - riceve i dati dal server e li ricompone
  - **visualizza le pagine Web (ipertesti),** mostrandone il contenuto e interpretando correttamente i linguaggi che vengono utilizzati per descriverne i contenuti

LE APPLICAZIONI WEB

- Programmare per il web significa scrivere due applicazioni in grado di comunicare tra loro:
  - un' applicazione client che invia richieste (e interpreta le risposte) e
  - un' applicazione server che riceve le richieste dell'applicazione client, prepara le risposte e le invia al client.
- Programmare per il web quindi implica necessariamente saper scrivere entrambi i tipi di applicazione.

PAGINE STATICHE E DINAMICHE

- Quando **ci connettiamo ad una risorsa in rete, identificata da un URL:**
  - Nel caso più semplice rappresenta l'indirizzo di una pagina (generalmente scritta in HTML) il cui **contenuto è fisso (Pagina STATICA);**
  - Nella maggior parte dei casi, l'URL punta all'indirizzo di una **pagina "dinamica"** il cui **contenuto viene generato (selezionato, composto) al momento della richiesta e/o elaborato al ricevimento;**

PAGINE WEB DINAMICHE

- **Nelle pagine Web "dinamiche" il contenuto** viene generato (selezionato,composto) al momento della richiesta o della visualizzazione
  - Pagine Web che vengono rese dinamiche con programmazione **client-side**
  - Pagine Web dinamiche che utilizzano programmazione **server-side**
  - Pagine Web dinamiche (la maggior parte) che utilizzano il concorso di entrambe le tecnologie.

PAGINE WEB DINAMICHE

- Client-side (Programmazione del Browser) :
  - JavaScript
  - Java Applet
  - Flash
- Server-side (Programmazione del Web Server):
  - ASP e ASP.NET di Microsoft
  - PHP
  - JSP (Java Server Pages)
- NB: Per le tecnologia Client-side è necessario che il Browser sappia interpretare le istruzioni!

HTML

FORMATTAZIONE...

- HTML è l'acronimo di HyperText Markup Language ("Linguaggio a marcatori per gli Ipertesti").
- Non è un linguaggio di programmazione non ha, cioè, meccanismi che consentono di prendere delle decisioni ("in questa situazione fai questo, in quest'altra fai quest'altro"), e non è in grado di compiere delle iterazioni ("ripeti questa cosa, finché non succede questo"), né ha altri costrutti propri della programmazione.
- Si tratta invece di un linguaggio descrittivo che usa dei contrassegni (o ' marcatori'), che permette di strutturare gli elementi di una pagina in blocchi la cui tipologia e le cui caratteristiche vengono definite attraverso appositi marcatori, detti **tag**.

… E STRUTTURA

- In origine HTML è stato concepito principalmente per formattare il testo: elementi con un preciso significato semantico si mischiavano con elementi di pura formattazione.
- Con l'evoluzione di html in xhtml e in html 5 i tag hanno sempre più assunto il compito di articolare la pagina i blocchi semantici e logici:
  - Tutti i tag di pura formattazione vanno considerati elementi deprecati.
  - Al contrario con html 5 sono stati introdotti tag che hanno l'unica funzione di definire in maniere più robusta le parti in cui è articolata una pagina Web (testata, piè di pagina, barra di navigazione, ecc.) dal punto di vista dell'organizzazione dei contenuti e dalla navigazione..
- Il compito di definire l'aspetto di una pagina è affidato ai fogli di stile.

I MARCATORI (TAG)

- I tag vanno inseriti tra parentesi uncinate: <TAG>
- La chiusura del tag viene indicata con una barra: </TAG>
- Il contenuto che il tag modifica va inserito tra l'apertura e la chiusura del tag medesimo:

```
Questa <span style="font-wieight:bold">parola</span> è in
                                grassetto.
```

- che nel rendering verrà reso:

Questa **parola** è in grassetto.

- Alcuni tag non hanno (o possono non avere) contenuto (empty tag) . Ad esempio l'interruzione di linea la indico così:

```
<br />
```

GLI ATTRIBUTI

- Le caratteristiche di un tag vengono determinate dagli attributi del tag. Ogni tag ha per i suoi attributi dei valori predefiniti che io posso modificare:

```
<TAG attributo_1="valore1" attributo_2="valore2">contenuto</TAG>
```

- Alcuni attributi sono generali, comuni a tutti i tag (id, class, ecc.), altri sono specifici:

```
<IMG width="20" height="20" src="miaImmagine.gif" alt="alt“ />
```

- Una caratteristica importante del codice HTML è che i tag possono essere annidati l'uno dentro l'altro.
- È quindi opportuno usare l'indentazione. Grazie ad essa il codice HTML risulta più leggibile.

CSS

REGOLE

- Un foglio di stile è costituito da una serie di regole che stabiliscono come un elemento (identificato da un selettore) viene reso su un media.

- Esempio:

```
p{
        font-family: Verdana, sans-serif;
        font-size:16px;
  }
```

SELETTORI

SELETTORI SEMPLICI

Selettore]	Es.	pref.	Descrizione	CSS
elemento	p		Selezione tutti gli elementi <p>	1
id	#nome	#	Seleziona l'elemento con id="nome"	1
*	*		Seleziona tutti gli elementi	1
classe	.intro	.	Seleziona tutti gli elementi con class="intro"	1
attributo	[title]		Seleziona tutti gli elementi che hanno l'attributo title	2
attributo=valore	[title="rosa"]		Seleziona tutti gli elementi che hanno l'attributo title="rosa"	2
attributo~=valore	[title~="rosa"]		Seleziona tutti gli elementi in cui l'attributo title contiene "rosa"	2
attributo|=valore	[title|="rosa"]		Seleziona tutti gli elementi in cui l'attributo title inizia con "rosa"	2

SELETTORI COMPOSTI

Selettore]	Es.	Descrizione	CSS
el1, el2, el3...	h1, h2, h3	Seleziona tutti gli elementi <h1>, <h2> e <h3>	1
el1 el2	div p	Seleziona gli elementi <p> contenuti in un elemento	1
el1 > el2	div > p	Seleziona gli elementi <p> il cui *parent* è un elemento	2
el1 + el2	h2 p	Seleziona gli elementi <p> che seguono immediatamente un elemento h2	2
.calsse el	.intro p	Seleziona gli elementi <p> contenuti in un elemento con class="intro"	1
#id el	#nome p	Seleziona gli elementi <p> contenuti nell'elemento con id="nome"	1

ecc.

PSEUDO-CLASSI

Selettore]	Es.	Descrizione	CSS
:link	a:link	Stato di link non visitato	1
:visited	a:visited	Stato di link visitato	1
:hover	a:hover	Aspetto quando il mouse è sopra l'elemento	1
:focus	input:focus	Aspetto quando l'elemento ha il *focus*	1
:first-letter	p:first-letter	Prima lettera del testo contenuto nell'elemento	2
:first-line	p:first-line	Prima riga del testo contenuto nell'elemento	2
:first-child	ul:first-child	Primo figlio di una elemento contenitore	2
:before e :after	div:after	Contenuto prima e dopo ad un elemento	2

JAVASCRIPT

COSA È JAVASCRIPT

- JavaScript è un linguaggio di programmazione.

COME FUNZIONA UN COMPUTER

- Un computer per poter risolvere un qualsiasi problema ha bisogno di qualcosa che gli indichi esattamente cosa fare passo dopo passo
- Un computer si limita ad eseguire delle istruzioni "*macchina*" per l'esecuzione delle quali e' stato progettato
- Un'istruzione "*macchina*" e' un'istruzione "primitiva" comprensibile e direttamente eseguibile dal calcolatore senza bisogno di essere interpretata

COME FUNZIONA UN COMPUTER

- Qualsiasi compito per poter essere eseguito da un calcolatore deve essere tradotto in un insieme di istruzioni *macchina*
- Un software non e' altro che un insieme di istruzioni eseguibili dal calcolatore che nel loro insieme risolvono un problema o eseguono un determinato compito
- E' compito del programmatore "tradurre" un problema utilizzando solamente il set di istruzioni "primitive" comprensibili al calcolatore

UN PROGRAMMA QUINDI È

- Una serie di istruzioni che il computer è in grado di eseguire
- Che elaborano DATI (cioè informazione correttamente codificate)
- Per risolvere un problema
- O meglio implementare un algoritmo

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- E' un linguaggio formale dotato di una sintassi ben definita che viene utilizzato per scrivere programmi che realizzano algoritmi.

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- Serve a facilitare la programmazione dei calcolatori rendendo possibile descrivere gli algoritmi e le strutture dei dati in una forma più vicina a quella del linguaggio umano scritto.

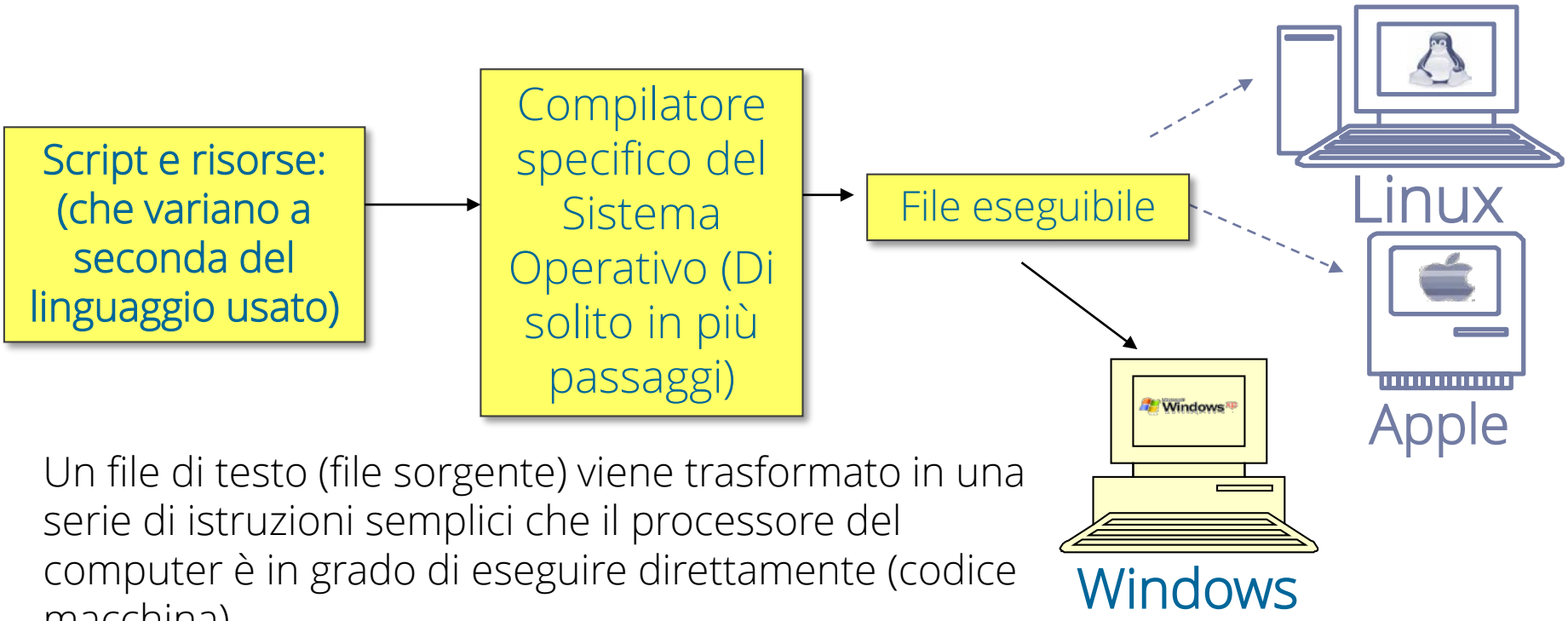
COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- A seconda del metodo utilizzato per tradurre il testo delle istruzioni in linguaggio macchina vengono suddivisi in due categorie: compilati (e semicompilati) e interpretati.

LINGUAGGI COMPILATI

- Il sorgente (un file di testo con le operazioni da eseguire) viene compilato in codice macchina e viene impacchettato in un particolare file detto eseguibile che il computer è in grado di eseguire direttamente senza bisogno di altro software;
- È specifico di un determinato sistema operativo e la compatibilità tra sistemi diversi può essere garantita solo dal fatto che esistano compilatori specifici per ogni sistema.

Linguaggi compilati

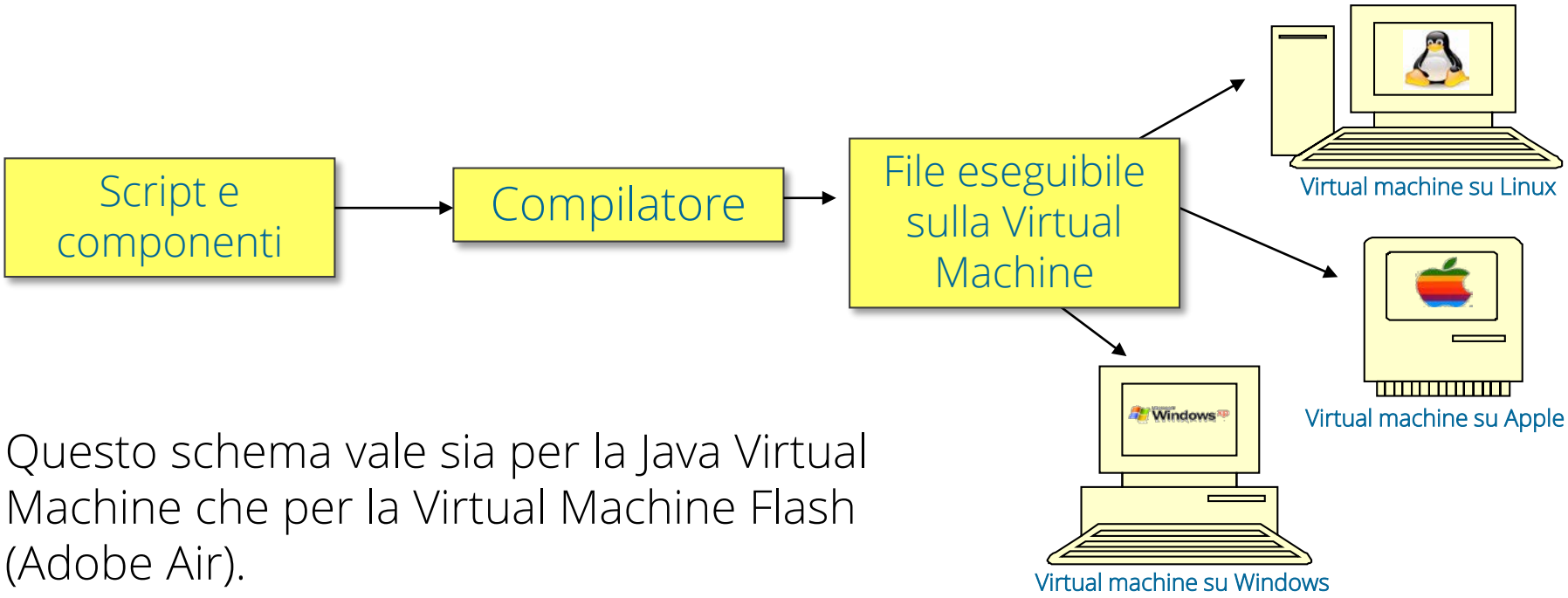


Un file di testo (file sorgente) viene trasformato in una serie di istruzioni semplici che il processore del computer è in grado di eseguire direttamente (codice macchina)

LINGUAGGI SEMICOMPILATI

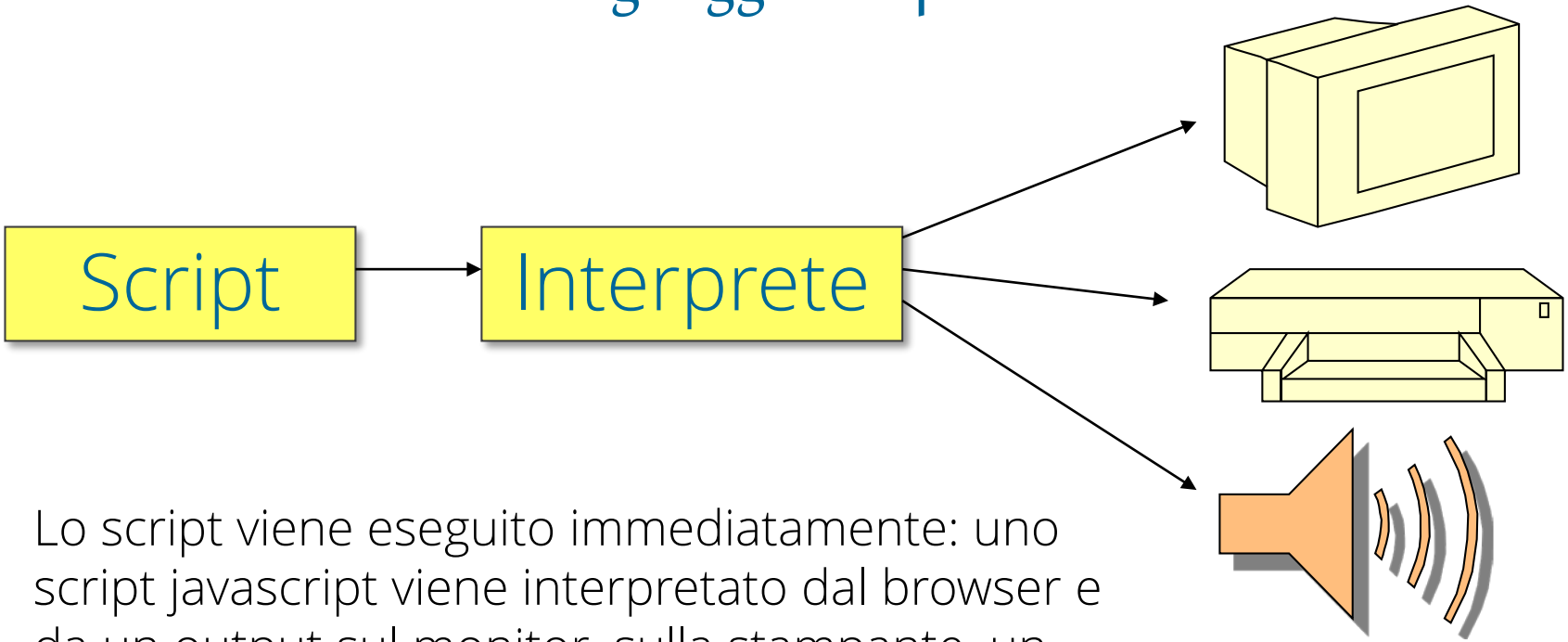
- Per semplificare la gestione della compatibilità non si compila il sorgente per uno specifico ambiente ma per una macchina virtuale.
- Per essere eseguiti i programmi presuppongono che la macchiana virtuale sia installata.

Linguaggi compilati



Questo schema vale sia per la Java Virtual Machine che per la Virtual Machine Flash (Adobe Air).

Linguaggi interpretati



Lo script viene eseguito immediatamente: uno script javascript viene interpretato dal browser e da un output sul monitor, sulla stampante, un output audio, ecc.

COMPILATO <> INTERPRETATO

- compilazione:
  - lo script viene elaborato dal compilatore prima di essere eseguito e la maggior parte degli errori di sintassi vengono individuati
- interpretazione:
  - Lo script viene eseguito così com'è, il controllo della correttezza del codice è affidato direttamente all'esecuzione dello stesso.

COSA È JAVASCRIPT

- Script in inglese significa "copione" o "sceneggiatura",.
- Il browser legge una riga, la interpreta e la esegue, poi passa alla successiva e fa la stessa cosa, e così di seguito fino alla fine dello script.
- Javascript è un linguaggio interpretato
- L'interprete utilizzato per eseguirlo è il browser

JAVASCRIPT NON È JAVA

- Con java si realizzano:
  - programmi (come StarOffice e OpenOffice)
  - applet (ma sono in disuso)
  - applicazioni lato server (J2EE, servlet, jsp...)
- Con JavaScript potete intervenire "solo" sulle vostre pagine web.

COSA È JAVASCRIPT

- è un linguaggio debolmente (o dinamicamente) tipizzato
- è in grado di reagire agli eventi.
  - Normalmente si scrive codice JavaScript da eseguire quando una pagina web ha terminato il caricamento, oppure quando un utente fa click su un determinato elemento HTML

COSA È JAVASCRIPT

- è in grado di leggere e scrivere gli elementi HTML.
  - Tramite JavaScript è possibile modificare la struttura del documento HTML in tempo reale, senza interagire con il server
- può essere utilizzato per convalidare i dati inseriti dall'utente prima di inviarli al server.

COSA È JAVASCRIPT

- può essere utilizzato per avere informazioni sul Browser del visitatore.
  - In questo modo possiamo decidere come comportarci a seconda del Browser che sta leggendo la pagina
- può essere utilizzato per creare i cookie e quindi archiviare e recuperare informazioni sul computer del visitatore

COME FUNZIONA

- è necessario comunicare al Browser sia la presenza di uno script sia il linguaggio in cui lo script è scritto.
- il meccanismo è simile a quello che abbiamo visto per i forgli di stile.

FOGLI INCORPORATI

- I fogli incorporati sono quelli inseriti direttamente nel documento (X)HTML tramite l'elemento <style>. Anche in questo caso la dichiarazione va posta all'interno della sezione <head>:

```
<html>
   <head>
       <title>Inserire i fogli di stile in un documento</title>
      <style type="text/css">
           body {
                 background: #FFFFCC;
            }
       </style>
    </head>
<body>
... .
```

IL TAG SCRIPT

- Per scrivere la sintassi è sufficiente aprire il tag <SCRIPT>. Il codice JavaScript va inserito tra l'apertura e la chiusura del tag. Così:

```
<script type="text/javascript">
  alert("ciao");
</script>
```

- Possiamo inserire il codice JavaScript in qualsiasi parte del documento (nella head oppure nel body) a seconda delle nostre esigenze.

FOGLI COLLEGATI

Uso dell'elemento <LINK>

- La dichiarazione va sempre collocata all'interno della sezione <HEAD> del documento (X)HTML:

```
<html>
  <head>
    . . . .
    <link rel="stylesheet" type="text/css"
  href="stile.css">
  </head>
<body>
  . . . .
```

FILE ESTERNO

- Quando si scrive codice di una certa lunghezza e/o che potrebbe essere ripetuto su più pagine
- Quando si utilizza un libreria Javascript esistente:

```
<script type="text/javascript" src="miofile.js"></script>
```

STILE IN LINEA

- L'ultimo modo per formattare un elemento con un foglio di stile consiste nell'uso dell'attributo 'style'.
- Esso fa parte della collezione di attributi (X)HTML definita Common: si tratta di quegli attributi applicabili a tutti gli elementi.
- La dichiarazione avviene a livello dei singoli tag contenuti nella pagina e per questo si parla di fogli di stile in linea. La sintassi generica è la seguente:

```
<elemento style="regole_di_stile">
```

GESTIONE DIRETTA EVENTO

- Come abbiamo detto Javascript è fatto principalmente per rispondere a degli eventi, come quello di un utente che clicca un elemento della pagina
- Si può associare direttamente del codice javascript all'evento di un elemento usando appositi attributi come onclick, onload, ecc:

```
<button onclick="alert('Ciao!')">Cliccami !</button>
```

No script

- All'interno del tag noscript può essere utilizzata la sintassi HTML per visualizzare messaggi:

```
<noscript>
<div align="center">
  <h3><font face="Verdana,Arial,Helvetica,sans-serif">
    Per visualzzare correttamente il contenuto della
    pagina occorre avere JavaScript abilitato.
  </font></h3>
</div>
</noscript>
```

jsfiddle.net

GLI ELEMENTI DEL LINGUAGGIO

INTRODUZIONE

- Istruzioni: parola riservata che il linguaggio usa per i comandi di base del linguaggio
- Variabile: nome simbolico a cui è associato un valore che può dipendere dall'input dell'utente e cambiare durante l'esecuzione del programma.
- Costante: quantità nota a priori che non dipende dall'input dell'utente e non cambia durante l'esecuzione del programma.
- Espressione: sequenza di variabili, costanti, espressioni collegate tra loro da operatori.

SINTASSI

- Ora prenderemo in esame questi elementi in termini grammaticali.
- Quello che diremo di JavaScript è sostanzialmente applicabile (salva variazioni di grammatica appunto) a tutti i linguaggi.

ELEMENTI DI UN LINGUAGGIO

- Le unità semantiche di base di ogni linguaggio sono:
  - *Parole chiave*
  - *Operatori e separatori*
  - *Letterali* (o *Costanti*)
  - *Nomi* (o *Identificatori*)

PAROLE CHIAVE

- Le parole chiave sono i termini (composti da caratteri alfanumerici), riservati al linguaggio di programmazione.
- Il creatore del linguaggio di programmazione stabilisce a priori quali termini riservare e quale sarà la loro funzione, il compito del programmatore è quello di impararle ed usarle in maniera appropriata.
- L'uso improprio di tali termini viene generalmente rilevato durante la fase di compilazione di un programma.

PAROLE CHIAVE IN JAVASCRIPT

```
break case catch const
continue debugger default
delete do else finally for
function if in instanceof new
return switch this throw try
typeof var void while with
```

Operatori

- Gli operatori sono token composti di uno o più caratteri speciali che servono a controllare il flusso delle operazioni che dobbiamo eseguire o a costruire *espressioni*
- Operatori usati sia in *JavaScript* che in *JAVA*:

```
++  !   !=  !==  %   %=   &   &&   &=  (  )  -   *   *=
,   .   ?:  /   //  /*  /=  []  ^   ^=  {}  |   ||  |=
~   +   +=  <   <<  <<=  <=  <>  =   -=  ==
===  >   >=  >>  >>=  >>>  >>>=
```

Proprietà degli operatori

- **Precedenza (o Priorità)**
  Indica l'ordine con il quale verranno eseguite le operazioni. Ad esempio in **4+7*5** verrà prima eseguita la moltiplicazione poi l'addizione.
- **Associtività**
  Un operatore può essere associativo a **sinistra** oppure associativo a **destra**. Indica quale operazione viene fatta prima a parità di priorità.

Separatori

- I separatori sono simboli di interpunzione che permettono di chiudere un'istruzione o di raggruppare degli elementi.
- Il separatore principale è lo *spazio* che separa i *termini* tra di loro quando non ci sono altri separatori. Gli altri separatori sono:

```
( ) { } , ; .
```

Letterali (o costanti)

- Le *costanti* (o letterali) sono quantità note a priori il cui valore non dipende dai dati d'ingresso e non cambia durante l' esecuzione del programma.
- La sintassi con cui le costanti sono descritte dipende dal tipo di dati che rappresentano.
- Le costanti servono:
  - A dare un valore iniziale ad una variabile
  - A confrontare un valore variable con un valore di riferimento

Costanti numeriche

- Le *costanti numeriche* iniziano sempre con un carattere numerico: il fatto che un *token* inizi con un numero basterà ad indicare al compilatore che si tratta di una costante numerica. Se il compilatore non potrà valutare quel *token* come numero segnalerà un errore.
- Il segno che separa la parte intera di un numero dalla parte decimale è il punto.
- È possibile inserire numeri in formato decimale, binario, ottale o esadecimale.
- Per segnalare al compilatore che un numero non è decimale si fa precedere il numero da un prefisso. Per i numeri esadecimali questo prefisso è 0x.
- Gli altri *termini* (*parole chiave* e *nomi*) NON possono iniziare con un numero.

Esempi di costanti numeriche

```
1
2433
1000000000
3.14
.33333333333
0.5
2345.675
0xFF0088
0x5500ff
0xff.00aa
```

Costanti stringa

- Una stringa è una sequenza di caratteri che permette di rappresentare testi. Un *costante* stringa è una sequenza (anche vuota) di caratteri racchiusi tra apici singoli o apici doppi.
- Per inserire ritorni a capo, tabulazioni, particolari caratteri o informazioni di formattazione si utilizzano speciali sequenze di caratteri dette *sequenze di escape*. Una sequenza di escape è formata da un carattere preceduto dal simbolo "\" (*backslash*). La sequenza di escape inserisce un carattere che non sarebbe altrimenti rappresentabile in un letterale stringa.

Principali sequenze di escape

\n nuova riga;

\r ritorno a capo;

\t tabulazione orizzontale;

\‘ apostrofo (o apice singolo);

\" doppio apice;

\\ backslash(essendo un carattere speciale deve essere inserito con una sequenza di escape).

Esempi di costanti stringa

```
// Stringa racchiusa da apici singoli
'Ciao a tutti'
// Stringa racchiusa tra apici doppi
"Ciao"
/* La sequenza di escape risolve l'ambiguità tra l'apostrofo
inserito nella stringa e gli apici singoli che la
racchiudono */
'Questo è l\'esempio corretto'
/* In questo caso non c'è ambiguità perché la stringa è
racchiusa tra doppi apici */
"Anche questo è l'esempio corretto"
/* Per inserire un ritorno a capo si usano le sequenze
di escape */
"Questa è una stringa valida\rdi due righe"
```

Costanti booleane

- Le costanti booleane, poiché rappresentano valori logici, possono avere solo due valori: vero (rappresentato dal letterale *true*) e falso (rappresentato dal letterale *false*).

Costanti di tipo Array

- Il letterale *Array* è costituito da una serie di elementi separati da virgole compresa tra due parentesi quadre:

```
// array che contiene i mesi dell'anno
["January", "February", "March", "April"];
```

Costanti di tipo Object

- Il letterale *Object* è invece compreso tra parentesi graffe ed è costituito da una serie di coppie "chiave:valore" separate da virgole:

```
//record di una rubrica telefonica in formato Object
{name:"Irving",age:32,phone:"555-1234"};
```

Identificatori (o Nomi)

- Un identificatore è un nome definito dal programmatore. Gli identificatori si usano per dare nomi alle variabili e alle funzioni.

Regole per gli Identificatori

- il primo carattere deve essere una lettera o il simbolo "_" (ricordiamo che nel caso la prima lettera fosse un numero il compilatore tenterebbe di interpretare il nome come costante numerica);
- i caratteri successivi possono essere lettere, numeri o "_".
- Gli identificatori non possono inoltre coincidere con le parole riservate del linguaggio.

VARIABILI

Pensiamo a quando salviamo un numero di telefono del nostro amico Mario sul cellulare; se vogliamo chiamare il nostro amico, basterà inserire il suo nome (Mario, nome della variabile) ed il cellulare comporrà automaticamente il numero di telefono (valore della variabile). Se per qualche ragione Mario cambierà numero di telefono, modificherò il contenuto della mia rubrica (cambierò il valore della variabile). In questa maniera senza modificare le mie abitudini (inserirò sempre Mario) il mio cellulare comporrà il nuovo numero.

VARIABILI

- Una variabile è composta da due elementi: il suo nome e il suo valore; come ho visto nell'esempio del cellulare in un programma posso usare i nomi delle variabili al posto dei valori che rappresentano.
- Ho la possibilità di usare simboli mnemonici al posto di numeri e stringhe di grande entità o difficili da ricordare.
- Ho la possibilità di usare il nome della variabile al posto del suo valore per eseguirvi sopra delle operazioni, e generalizzare l'elaborazione.

variabili

- Prima di usare una variabile la dichiaro usando l'istruzione var.
- Per assegnare alla variabile un valore utilizzo l'operatore di assegnazione ("=").

```
// creo una variabile che si chiama "mioNome"
var mioNome;

//assegno a mioNome il contenuto “Pippo"
mioNome="Pippo";
```

TIPI IN JAVASCRIPT

Tipo di dati	Spiegazione	Esempio
Number	Qualsiasi valore numerico	miaVariabile=300;
Number	Numeri con virgola	miaVariabile=12.5;
String	Qualsiasi valore letterale. È una sequenza di caratteri, racchiusa tra virgolette.	miaVariabile="Wolfgang";
Null	È uno speciale tipo di dato che indica l'assenza di alcun valore ("è il nulla"). Non è lo zero.	miaVariabile=null;

Boolean	È uno tipo di dato che indica uno stato. Di fatto un valore booleano può assumere solo due valori: acceso (vero), spento (falso). È il classico "interruttore della luce".	//Vero:
		miaVariabile=true;
		//Falso:
		miaVariabile=false;

Object	Array (Elenco di valor)i	miaVariabile=['lunedì', 'martedì', 'mercoledì', 'giovedì', 'venerdì', 'sabato', 'domenica']
Object	Informazione complessa	miaVariabile = {nome:"Mario", cognome:"Rossi", eta:25}

eventi

evento	si applica a...	esempio
onload	<body>, <img>	<body onload="alert('ciao');">
onunload	<body>	<body onunload="alert('ciao');“>
onmouseover	<a>, <area>, <input> (submit, button, ecc.)	<a onmouseover="alert('ciao');" href="pagina.html">
onmouseout	<a>, <area>, <input>	<a onmouseout="alert('ciao');" href="pagina.html">
onclick	<a>, <area>, <input>	<a onclick="alert('ciao');" href="pagina.html">
onkeypress	<a>, <area>, <input>,	<textarea onkeypress="alert('ciao');“></textarea>
onchange	<select>	<select onchange="alert('ciao');“> <option>uno </option> ..... </select>
onsubmit	<form>	<form name="mioform" action="http://...." onsubmit="alert('ciao');">
onfocus	<a>, <input>, <body>	<body onfocus="alert('ciao');“>
onblur	<a>, <input>, <body>	<body onblur="alert('ciao');">