

```
TMLElement ptr1 =  
("a[href='https://www.kinderc.  
ptr1.className = "link";  
ptr1.style["color"] = "violet";  
typedef HTMLMediaElement player;  
fetch("/songs-endpoint.php", {"POST",  
{"reqInfo\":"songs\"}})  
.then([](Request& response) {  
    if(response.status == 200) {  
        var objSong = response.JSON();  
        int songNumber;  
        var  
HTMLMediaElement
```

#kinderc th"];

Sviluppare WebApp funzionali utilizzando C++.

```
objSong[0]["audioURL"];  
aud  
playerPtr->attributes["src"] =  
} else $("dialog.errorbox").innerHTML = R"(  
<h1>Sembra che ci sia un errore.</h1>  
<br />  
<div>  
    La canzone che hai richiesto pot  
    essere più disponibile. Controlla oppure  
    più tardi.  
</div>
```

Sommario

Introduzione	1
Che cos'è KinderC	1
Premesse per il programmatore	1
Link Utili	2
Repository GitHub del progetto	2
Compilatore CLANG	2
Impostazione dell'ambiente di lavoro	2
Struttura di un'applicazione KinderC	2
Hello World	3

Introduzione

Che cos'è KinderC

KinderC non è altro che una libreria per il linguaggio C++ che permette, attraverso l'utilizzo del compilatore CLANG, di realizzare siti e applicazioni web snelle e performanti. Il codice scritto dal programmatore viene compilato in un file binario in formato wasm, utilizzando la tecnologia WebAssembly, che viene poi incluso in una pagina HTML e mandato in esecuzione al suo caricamento. Presenta alcuni vantaggi rispetto all'utilizzo di JavaScript standard, che qui elenchiamo:



- **Velocità:** Essendo compilata, un'applicazione KinderC tende ad essere più veloce rispetto ad una WebApp scritta in JavaScript standard.
- **Efficienza:** Servendosi di un linguaggio a tipizzazione forte, il programmatore può decidere di allocare variabili grandi quanto meglio crede. Può inoltre allocare la memoria in maniera dinamica.
- **Sicurezza:** Il codice scritto viene compilato in un file WebAssembly binario. Risulta quindi abbastanza complesso andare a decompilarlo per ritornare al codice sorgente, cosa che rappresenta un vantaggio in termini di sicurezza.
- **Semplicità:** Essendo JavaScript un linguaggio C-like, è facile imparare a scrivere codice utilizzando KinderC. Inoltre, tantissimi metodi richiamano le librerie C standard oppure i metodi propri di JavaScript client-side.

Premesse per il programmatore

Il programmatore che intende imparare a utilizzare la libreria dovrebbe avere queste conoscenze di base:

- **Conoscenza del linguaggio C/C++** e della sua sintassi, almeno a livello superficiale.
- **Conoscenza di HTML, CSS, JavaScript.**

Se così non fosse, si consiglia di consultare guide sul linguaggio C (ed eventualmente C++, se si intende programmare a oggetti) prima di proseguire.

Link Utili

Repository GitHub del progetto

Questa guida, insieme al file header e alle implementazioni di tutti i metodi della libreria, sono pubblicamente disponibili su GitHub a questo link: <https://github.com/nboano/kinderc>

Compilatore CLANG

Per compilare i nostri codici, ci serviremo del compilatore clang, scaricabile a questo link:

<https://releases.lldvm.org/download.html>

Al fine che tutto funzioni completamente, è necessario che l'eseguibile di clang sia aggiunto al PATH di sistema.









Impostazione dell'ambiente di lavoro

Per sviluppare utilizzando la libreria è necessario scaricarla dal repo GitHub, il cui link si trova nel paragrafo precedente. Fatto ciò, è necessario anche scaricare e installare il compilatore clang.

N.B. Si consiglia di clonare il repository in una cartella di facile accesso, specie per semplificare le operazioni di inclusione della libreria e di compilazione da terminale. In tutti gli esempi di questo manuale, il repository sarà stato clonato nella cartella **D:\kinderc**. Si adattino i comandi in relazione a dove si è scelto di collocare il file header e lo script per la compilazione.

Per scrivere codice è possibile utilizzare un qualsiasi editor di testo. Tuttavia, si raccomanda l'uso di **Visual Studio Code**, particolarmente leggero e con un'evidenziazione della sintassi ricca e capibile.

Nell'immagine è possibile vedere i file della libreria, che dovrebbero essere presenti sulle vostre macchine prima di iniziare a lavorare.

 .git	20/11/2022 21:34	Cartella di file	
 .vs	20/11/2022 21:34	Cartella di file	
 code	19/11/2022 02:22	Cartella di file	
 .gitattributes	16/11/2022 15:23	File GITATTRIBUTES	3 KB
 .gitignore	16/11/2022 15:23	File GITIGNORE	7 KB
 kccompil.bat	17/11/2022 21:45	File batch Windows	1 KB
 kinderc.hpp	20/11/2022 21:20	C/C++ Header	33 KB
 kinderc.js	19/11/2022 11:14	JavaScript File	2 KB

Struttura di un'applicazione KinderC

La WebApp di compone fondamentalmente di tre parti:

- **UN FILE HTML**, di solito nominato *index.html*, che contiene la struttura della pagina, e che nel tag head presenta un riferimento agli altri file.
- **UNO SCRIPT *kinderc.js***, soprannominato "glue code". Esso si trova nella libreria da voi scaricata e ha il compito di mettere in comunicazione l'interprete JavaScript, e quindi il DOM stesso, con il compilato da voi scritto.

La cosa più comoda per includere il *kinderc.js* è utilizzare la CDN:

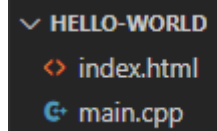
<https://cdn.jsdelivr.net/gh/nboano/kinderc/kinderc.js>

- **UN FILE SORGENTE *main.cpp*, che verrà compilato in un file binario *main.wasm*, che contiene effettivamente il codice.**

Hello World

Cominciamo quindi con un primo esempio, un classico Hello World.

Creiamo una cartella sul nostro computer a cui possiamo accedere da browser, utilizzando un WebServer a nostra scelta. È importante aprire le applicazioni passando da un WebServer e non semplicemente cliccando sul file index.html, in quanto il browser non permette di richiedere il file wasm quando aperto così.



All'interno della cartella, creiamo un nuovo file index.html, insieme ad un file main.cpp.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>KinderC Hello World</title>

  <script src="https://cdn.jsdelivr.net/gh/nboano/kinderc/kinderc.js"></script>
  <assembly src="main.wasm"></assembly>
</head>
<body>

</body>
</html>
```

Nel file HTML devono essere specificati il percorso del futuro file wasm compilato, e anche quello del file kinderc.js, come da esempio.

Creiamo quindi anche il file sorgente, main.cpp, su cui andremo a codificare il nostro primo esempio.

```
#include "D:\kinderc\kinderc.hpp"

int main() {
    printf("<h1>Hello World!</h1>");
}
```

Nel file sorgente C++ includiamo la libreria precedentemente scaricata.

Ora è il momento di compilare. Apriamo un terminale (anche quello interno di VSCode), e spostiamoci nella cartella del progetto.

Lanciamo il seguente comando per compilare:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

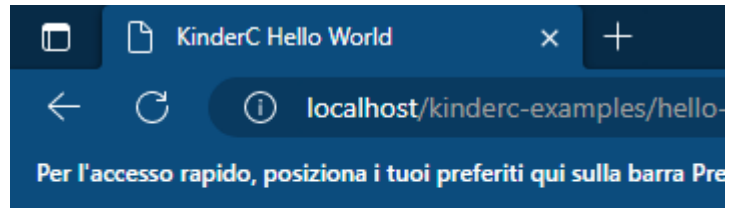
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti riservati.

Installa la versione più recente di PowerShell per nuove funzionalità e miglioramenti. https://aka.ms/PSWindows

PS D:\IIS\www\kinderc-examples\hello-world> D:\kinderc\kccompile main.cpp main.wasm
PS D:\IIS\www\kinderc-examples\hello-world> |
```

Ovviamente il comando va adattato a seconda di dove avete deciso di collocare il file *kccompile.bat*.

Se non ci sono errori, il tutto dovrebbe funzionare.



Hello World!