JavaScript to jQuery converter

MANUALE UTENTE

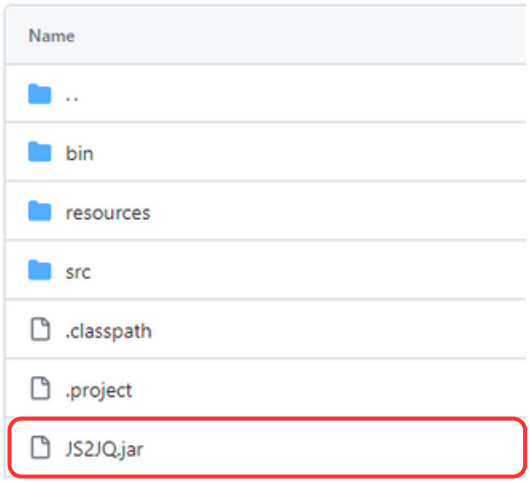
Jacopo Boffelli [1053217]
Mauro Riva [1053644]
A.A. 2022/2023

Installazione

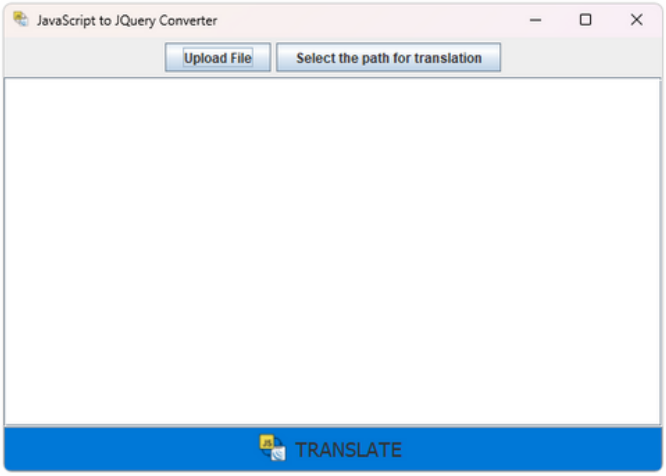
L'applicazione è disponibile per il download sulla piattaforma github al seguente link:

https://github.com/MauroRiva98/JavaScriptToJQueryConverter.git

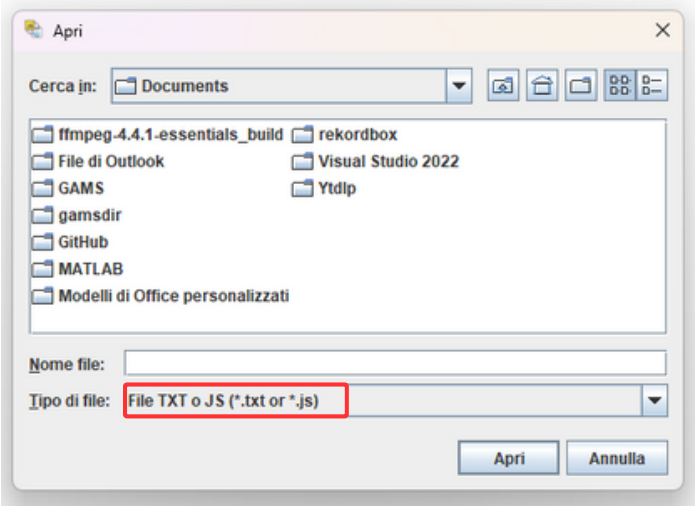
Nella cartella JS2JQ si trova il file runnable JAR JS2JQ.jar con il quale è possibile avviare l'applicazione.



Istruzioni per l'uso

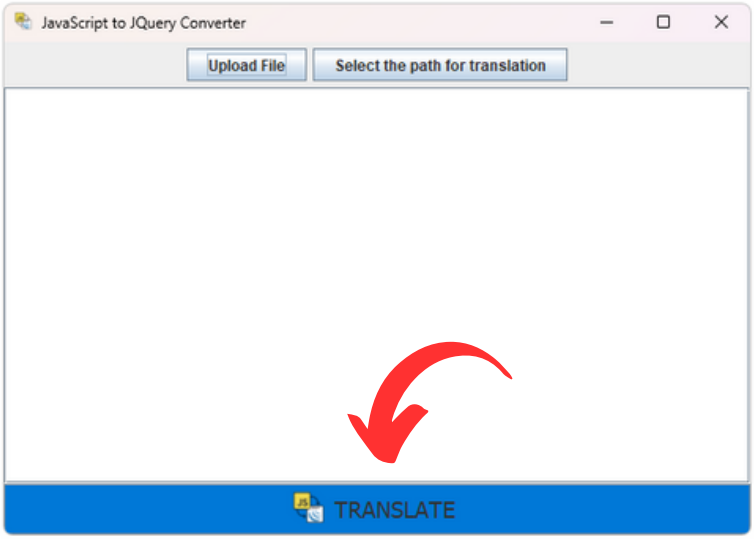


La figura rappresenta l'interfaccia all'avvio dell'applicazione nella quale è possibile attraverso due pulsanti selezionare il file di input per la traduzione e impostare la directory e il nome del file di output

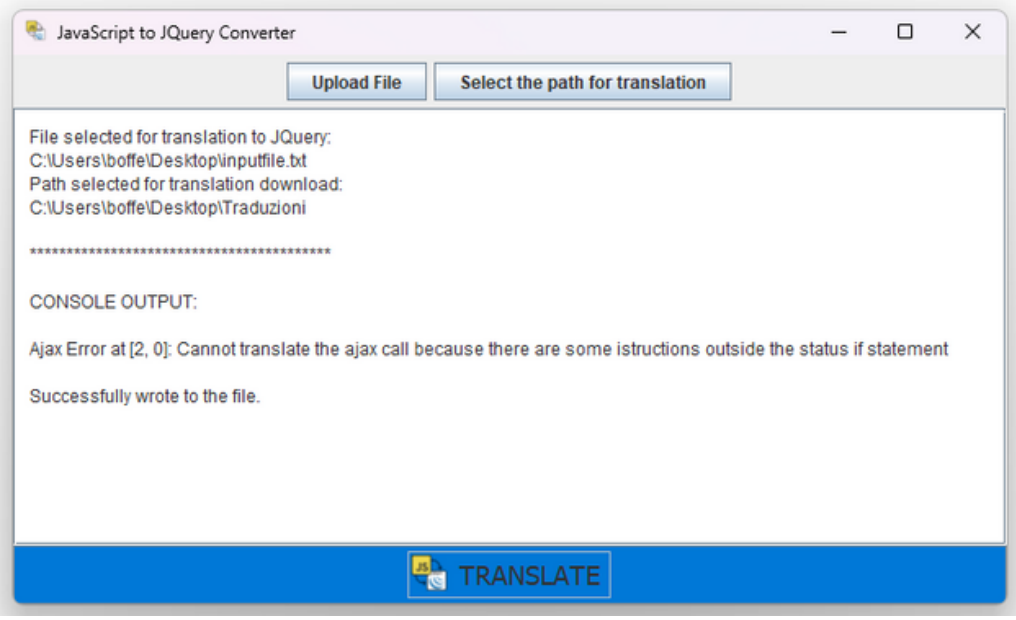


Nella selezione del file di input è possibile selezionare solamente file in formato txt e file di tipo js.

Nel caso di mancata selezione della directory dove salvare il file convertito viene selezionata di default la cartella nella quale si trova il file eseguibile dell'applicazione utilizzando come nome di default "translated", in questo caso il file manterrà il formato originario.



Il pulsante "translate" avvia la traduzione del file selezionato. Eventuali segnalazioni o comunicazioni vengono visualizzati attraverso la console.



Funzionalità del progetto

L'applicazione permette la conversione di un codice scritto in JavaScript nativo nel suo corrispettivo implementando jQuery secondo le seguenti modalità e vincoli.

- **Manipolazione DOM**

Sono convertite le operazioni di manipolazione della pagina web, quali:

document.getElementById("test") → $("#test")

document.getElementsByName("test") → $("[name=test]")

document.getElementsByClassName("test") → $(".test")

document.getElementsByTagName("test") → $("test")

- **Attributi JavaScript**

HTML

object.innerHTML → object.html()

object.innerHTML = "<h1> Titolo </h1>" → object.html("<h1> Titolo </h1>")

object.innerHTML += "<h1> Titolo </h1>"

→ object.html(object.html() + "<h1> Titolo </h1>")

VAL

object.value → object.val()

object.value = "Mario Rossi" → object.val("Mario Rossi")

object.value+= "Mario Rossi"

→ object.val(object.val() + "Mario Rossi")

TEXT

object.textContent → object.text()

object.textContent = "Europa" → object.text("Europa")

object.textContent += "Europa"

→ object.text(object.text() + "Europa")

CSS

object.style.top → object.css("top")

object.style.top = "100px" → object.css("top", "100px")

HIDE

object.display = "none" → object.hide()

SHOW

object.display = "block" → object.show()

- **Funzioni JavaScript**

ATTR

object.getAttribute("src") → object.attr("src")

ADDCLASS

object.classList.add("mystyle") → object.addClass("src")

- **Chiamate AJAX**

STATUS NON SPECIFICATI

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
        const xmlDoc = xhttp.responseXML;
        const cd = xmlDoc.getElementsByTagName("CD");
        myFunction(cd);
}
xhttp.open("GET", "cd_catalog.xml");
xhttp.send();
```

```
$.ajax({url:"cd_catalog.xml", type: "GET",
        success: function(data, textStaus, jqXHR) {
        const xmlDoc = jqXHR.responseXML;
        const cd = xmlDoc.getElementsByTagName("CD");
        myFunction(cd);
}});
```

STATUS SPECIFICATI

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
     if(this.status == 200){
          console.log("ok");
    }
     else if (this.status == 404){
          console.log("error");
    }
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

```
$.ajax({url:"file.txt",
          type: "GET",
          statusCode: {404: function(data, textStatus, jqXHR)
                              {console.log("error"); },
                              200: function(data, textStatus, jqXHR)
                              {console.log("ok"); }}
});
```

VINCOLI

Se nelle chiamate AJAX i seguenti vincoli non sono rispettati queste non saranno convertite in jQuery ma rimarranno invariate nel file di output.

1. STATUS SPECIFICATI SENZA ISTRUZIONI ALL'ESTERNO DEL BLOCCO IF

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
    console.log("istruzione esterna");
    if(this.status == 200){
        console.log("ok");
    }
    else if (this.status == 404){
        console.log("error");
    }
    console.log("istruzione esterna");
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

2. UN SOLO BLOCCO DI CODICE ASSEGNATO AD UNO STATUS

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
    if(this.status == 200){
        console.log("ok");
    }
    if(this.status == 200){
        console.log("successo");
    }
    else if (this.status == 404){
        console.log("error");
    }
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

3. UN'UNICA OPERAZIONE LOGICA PER CONDIZIONE

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
    if(this.status == 200 && flag==false){
        console.log("ok");
    }
    else if (this.status == 404){
        console.log("error");
    }
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

4. ASSENZA DELL'ELSE NEGLI IF DELLO STATUS

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
    if(this.status == 200){
        console.log("ok");
    }
    else if (this.status == 404){
        console.log("error");
    }
    else console.log("altro");
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

ASSUNZIONI

Un' assunzione fatta durante lo sviluppo è stata considerare la chiamata AJAX come un blocco contiguo senza istruzioni aggiuntive nel mezzo, ad eccezione dell'inizializzazione dell'oggetto di tipo XMLHttpRequest che si può trovare in una qualsiasi posizione del codice.

```
const xhttp = new XMLHttpRequest();
console.log("istruzione esterna");
xhttp.onload = function(){
     if(this.status == 200){
          console.log("ok");
     }
     else if (this.status == 404){
          console.log("error");
     }
}
xhttp.open("GET", "file.txt");
xhttp.send();
```

```
const xhttp = new XMLHttpRequest();
xhttp.onload = function(){
     if(this.status == 200){
          console.log("ok");
     }
     else if (this.status == 404){
          console.log("error");
     }
}
xhttp.open("GET", "file.txt");
console.log("istruzione esterna");
xhttp.send();
```

Perdita di informazioni!