



**UNIVERSITÀ
DEGLI STUDI
DI UDINE**

hic sunt futura

Tecnologie Web e Laboratorio (a.a. 2021-2022)

JavaScript, Web Application Programming Interface (API), Web Application Framework

Daniele Salvati

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Università degli Studi di Udine

daniele.salvati@uniud.it

DISCLAIMER

- Non è consentita la diffusione, ma solo l'utilizzo per lo studio personale

Programmazione del Web

- **HTML** -> definisce il contenuto e la struttura delle pagine web
- **CSS** -> specifica i fogli di stile delle pagine web



Web Statico

Programmazione avanzata del Web

Web Dinamico (Web 2.0)



- **Programmazione lato client:** operazioni di elaborazione effettuate da un client in un'architettura client-server
- **Programmazione lato server:** operazioni di elaborazione compiute dal server in un'architettura client-server

Web dinamico

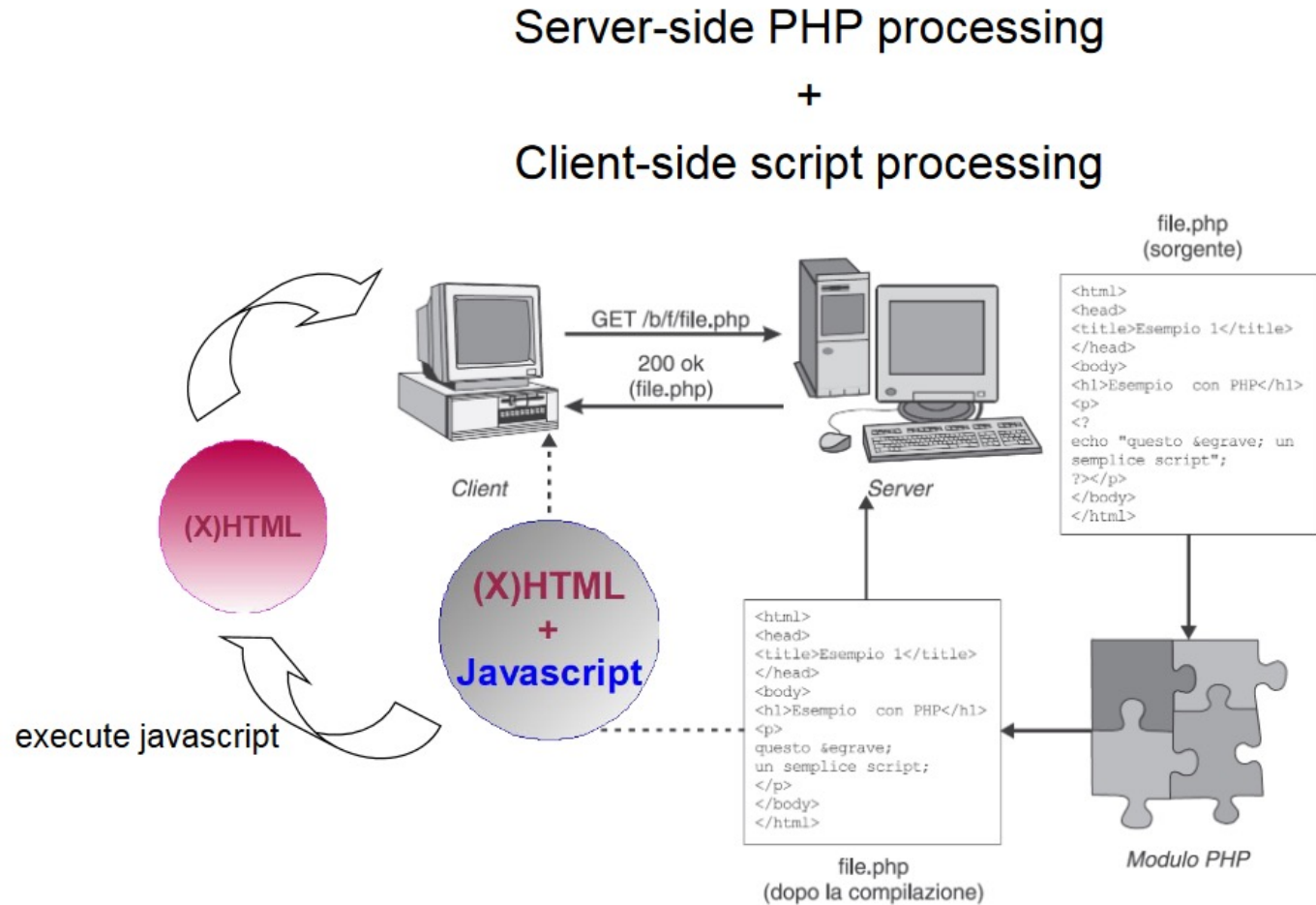
- Le tecnologie avanzate per lo sviluppo del Web 2.0 dinamico possono essere classificate in tre categorie:
 - **Linguaggi di programmazione e Scripting**
 - **Web Application Programming Interface (API)**
 - **Web Application Framework**

LINGUAGGI DI PROGRAMMAZIONE E SCRIPTING - JAVASCRIPT

Linguaggi di programmazione e Scripting

- Linguaggi più diffusi:

- JavaScript
- PHP
- SQL
- Python
- JAVA



JavaScript


- **JavaScript** è il linguaggio di programmazione più diffuso al mondo ed è «**il linguaggio di programmazione del Web**»
- **JavaScript** è un linguaggio di scripting, progettato per essere ospitato ed eseguito all'interno altri programmi
- Il tipico programma host per JavaScript è il browser (il browser ha un interprete JavaScript che esegue lo script quando la pagina viene visitata)

JavaScript (2)

- L'interfaccia che consente a JavaScript di interagire con il browser si chiama **DOM (Document Object Model)**
- I siti Web possono utilizzare la tecnologia **JavaScript lato client** per creare elementi dinamici nelle applicazioni Web
- Il ruolo principale delle applicazioni Web Javascript è la progettazione di funzioni integrate nei documenti HTML, che interagiscono con il browser DOM per completare azioni come controllare i campi di input, nascondere o mostrare elementi, aggiornare parti della pagina Web

JavaScript (3)

- **JavaScript** è un linguaggio di programmazione orientato agli oggetti e agli eventi
- Ad esempio un oggetto che modella una *car*, definendo alcune proprietà e alcuni metodi

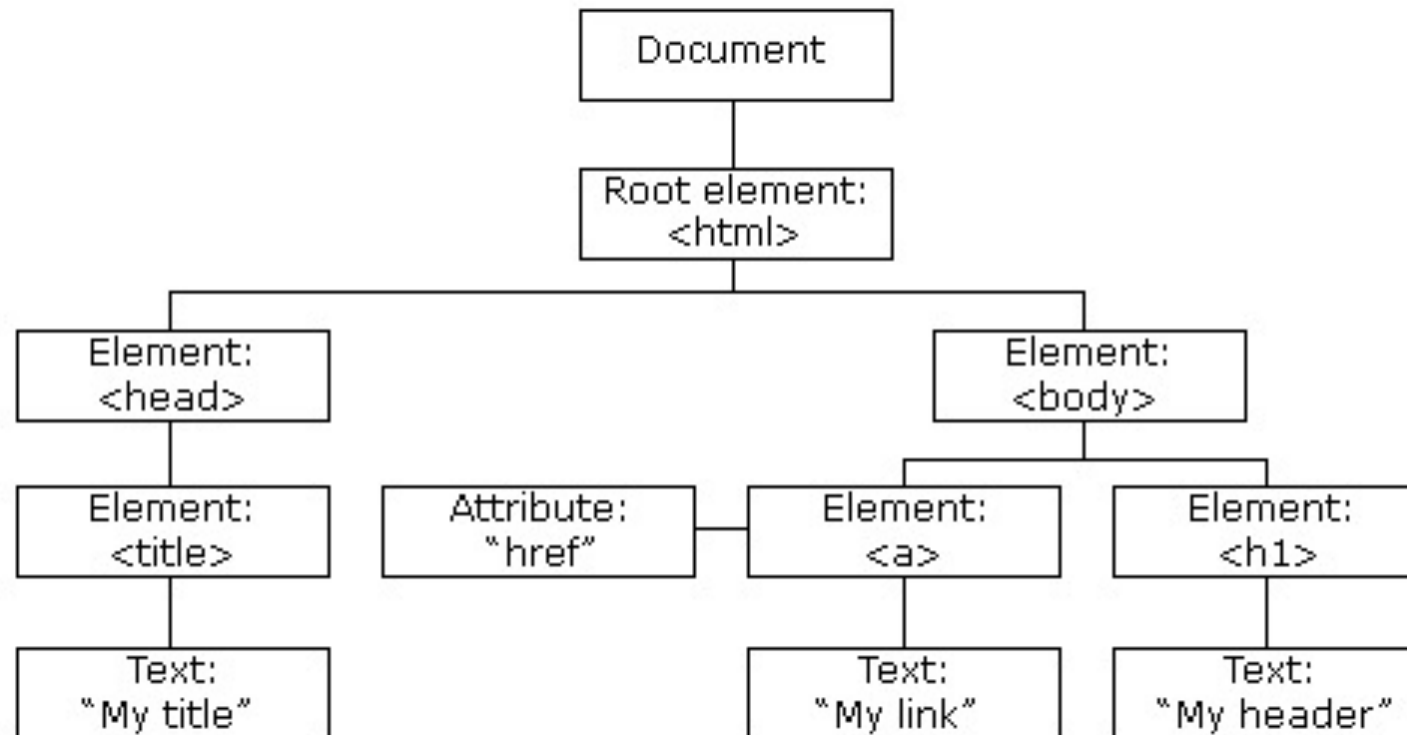
| Object | Properties | Methods |
|--|---|---|
|  | <code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code> | <code>car.start()</code> <code>car.drive()</code> <code>car.brake()</code> <code>car.stop()</code> |

JavaScript (4)

- Con il **modello a oggetti**, JavaScript può effettuare una serie di operazione su una pagina Web per creare **HTML dinamico**
 - **modificare tutti gli elementi HTML**
 - **modificare tutti gli attributi HTML**
 - **cambiare tutti gli stili CSS**
 - **rimuovere elementi e attributi HTML esistenti**
 - **aggiungere nuovi elementi e attributi HTML**
 - **reagire a tutti gli eventi HTML esistenti**
 - **creare nuovi eventi HTML**

HTML DOM (Document Object Model)

- Il modello HTML DOM è costruito come un albero di oggetti



Esempio JavaScript

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!">Click Me!</button>

</body>
</html>
```



What Can JavaScript Do?

JavaScript can change HTML content.

Click Me!



Dopo il click sul pulsante

What Can JavaScript Do?

Hello JavaScript!

Click Me!

JavaScript (5)

- In HTML, il codice JavaScript è inserito tra i tag **<script>** e **</script>** che può essere messo nel **<head>** o nel **<body>** del documento **HTML**
- Il codice JavaScript può essere scritto anche su un file separato identificato da un nome seguito dal suffisso **.js** e inserito nel HTML con l'attributo **src**

- Esempio

<script src="myScript.js"></script>

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>
```

Accede ad un
elemento HTML

Scrive in un
elemento HTML



GLI ELEMENTI DI BASE DEL LINGUAGGIO

Commenti, punti e virgola e maiuscole

- I commenti si inseriscono con
 - `//commento per singola riga`
 - `/* commento multiriga */`
- Ogni istruzione (o blocco di istruzioni) è delimitata da un *punto e virgola* (;) (non è obbligatorio ma è buona norma inserirlo)

- **Esempio**

```
<script>  
var x;  
x = 6;  
</script>
```

- JavaScript è ***case sensitive***: fa distinzione tra maiuscole e minuscole nei nomi di istruzioni, variabili e costanti, ecc.

Tipi di dato

- JavaScript prevede cinque tipi di dato primitivi
 - Numeri
 - Stringhe
 - Booleani
 - Null
 - undefined

Esempio

```
<script>  
var x;  
x = 6.2;  
x = null;  
var nome;  
nome = "Mario";  
var loop;  
loop = true;  
</script>
```

Variabili, operazioni ed espressioni

- Le variabili sono usate per memorizzare valori o oggetti durante l'esecuzione degli script
- Le variabili possono essere create con **const**, **var**, e **let** (non può essere ridichiarata)
- Esempio

```
<script>  
const pigreco = 3.14;  
var x;  
let y;  
</script>
```

- Operazioni ed espressioni sono usati per modificare le variabili

Array

- Un array è una variabile speciale, che può contenere più di un valore

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Arrays</h2>

<p id="demo"></p>

<script>
const cars = [
  "Saab",
  "Volvo",
  "BMW"
];
document.getElementById("demo").innerHTML = cars[0];
</script>

</body>
</html>
```



JavaScript Arrays

Saab

CONTROLLO DI FLUSSO

Istruzioni per stabilire il flusso di esecuzione

- **If**: istruzione condizionale
- **If-else e switch-case**: istruzione con più condizioni
- **While e do-while**: istruzioni per iterazioni di base
- cicli **for**
- **Break e continue**

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>
const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];

let text = "";
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```



JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

Esempio

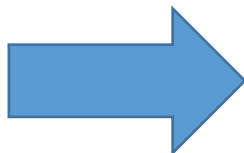
```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript switch</h2>

<p id="demo"></p>

<script>
let day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
  }
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

</body>
</html>
```



JavaScript switch

Today is Tuesday

LE FUNZIONI

Function

- Una funzione JavaScript è un blocco di codice progettato per eseguire un'attività particolare
- **Sintassi**

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

```
<!DOCTYPE html>  
<html>  
<body>  
<p id="demo"></p>  
<script>  
var x = myFunction(4, 3);  
document.getElementById("demo").innerHTML = x;  
function myFunction(a, b) {  
    return a * b;  
}  
</script>  
</body>  
</html>
```



12

Esempio

- Le variabili dichiarate all'interno di una funzione sono visibili solo all'interno della funzione

```
<!DOCTYPE html>
<html>
<body>

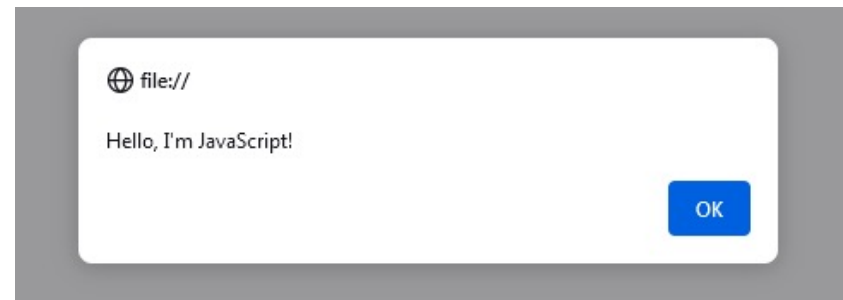
<script>
function showMessage() {
  let message = "Hello, I'm JavaScript!"; // variabile locale

  alert( message );
}

showMessage(); // Hello, I'm JavaScript!

alert( message ); // <-- Errore! La variabile è locale alla funzione
</script>

</body>
</html>
```



Window BOM (Browser Object Model)

- Il Browser Object Model (BOM) è il modello gerarchico che rappresenta l'oggetto browser e consente a JavaScript di "parlare con" il browser
- **window** è l'oggetto globale (padre di tutti gli altri oggetti) che rappresenta una finestra aperta
- Tutti gli altri oggetti riflessi dal browser sono proprietà dell'oggetto window
- Alcuni metodi
 - [window.]**alert("messaggio")**: visualizza una finestra di messaggio
 - [window.]**close()**: chiude la finestra
 - [window.]**open("url", "nome", "lista_opzioni")**: apre una nuova finestra

Eventi

- Un evento consiste nella notifica di un avvenimento specifico che si è verificato all'interno del browser o in seguito a un'azione dell'utente

Handler degli eventi

| | | |
|-----------------------------------|--|--|
| Passaggio del mouse | onMouseOver onMouseOut | il mouse si muove sull'oggetto il mouse si allontana dall'oggetto |
| Tasti del mouse | onClick onDbClick onMouseDown onMouseUp onDragDrop | l'utente fa click l'utente fa doppio click è premuto il tasto sx è rilasciato il tasto sx avviene un trascinamento |
| Modifiche dell'utente | onChange | l'utente modifica il valore di un campo-form |
| Eventi legati al fuoco | onFocus onBlur onSelect | l'oggetto riceve il fuoco l'oggetto perde il fuoco avviene una selezione (mouse o shift+frecce) |
| Caricamento degli oggetti | onLoad onUnload | l'oggetto viene caricato l'oggetto viene scaricato |
| Eventi legati alle finestre | onResize onScroll | la finestra viene ridimensionata avviene uno scroll |
| Eventi legati a pulsanti speciali | onSubmit onReset | l'utente fa click sul pulsante Submit l'utente fa click sul pulsante Reset |

- Per approfondire le funzionalità del JavaScript: <https://www.w3schools.com/js/>

LE CLASSI

Class

- Una classe Javascript è un **template** per un oggetto
- Quando si definisce una classe, si usa la classe per creare oggetti
- **Sintassi**

```
class ClassName {  
  constructor() { ... }  
  method_1() { ... }  
  method_2() { ... }  
  method_3() { ... }  
}
```

Esempio

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Class Method</h2>

<p>How to define and use a Class method.</p>

<p id="demo"></p>

<script>
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;

  }
  age() {
    let date = new Date();
    return date.getFullYear() - this.year;
  }
}

let myCar = new Car("Ford", 2014);
document.getElementById("demo").innerHTML =
"My car is " + myCar.age() + " years old.";
</script>

</body>
</html>
```



JavaScript Class Method

How to define and use a Class method.

My car is 7 years old.

WEB APPLICATION PROGRAMMING INTERFACE (API)

Web API

- Una Web API è un'interfaccia per la programmazione di applicazioni per il Web
- Un'API del browser può **estendere le funzionalità** di un browser web
- Un'API server può estendere le funzionalità di un server web
- Tutti i browser dispongono di una serie di Web API integrate per supportare **operazioni complesse** e per **facilitare** l'accesso ai dati
- Le API sono uno strumento molto utile per gli sviluppatori perché forniscono una modalità di interazione tra componenti software per inserire **funzioni avanzate nelle applicazioni web**

Web API (2)

- Dall'inizio degli anni 2000 le API sul Web hanno avuto una crescita esponenziale, sia in termini di numerosità delle interfacce esposte, sia di varietà dell'ambito applicativo:
 - servizi finanziari/di pagamento
 - social media
 - distribuzione di contenuti
 - mapping
 - contenuti multimediali
 - streaming in tempo reale



Web API (3)

- Le Web API sono in genere utilizzate con JavaScript, anche se possono essere usati altri linguaggi
- Alcune API
 - **Canvas**: disegno grafico
 - **Geolocation**: geo localizzazione dei dispositivi
 - **Media Capture and Streams**: gestione degli stream
 - **Web Audio**: programmazione avanzata dell'audio nel web
 - **WebGL**: grafica 2D e 3D interattiva
 - **WebRTC**: streaming audio/video e altri dati
 - **XMLHttpRequest**: utilizzato per interagire con i server

Lista completa della API Web supportate dai browser più diffusi: <https://developer.mozilla.org/en-US/docs/Web/API>

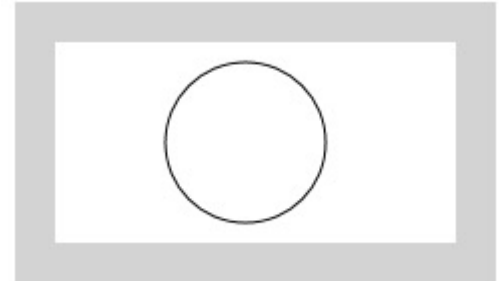
Esempio Canvas

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:20px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

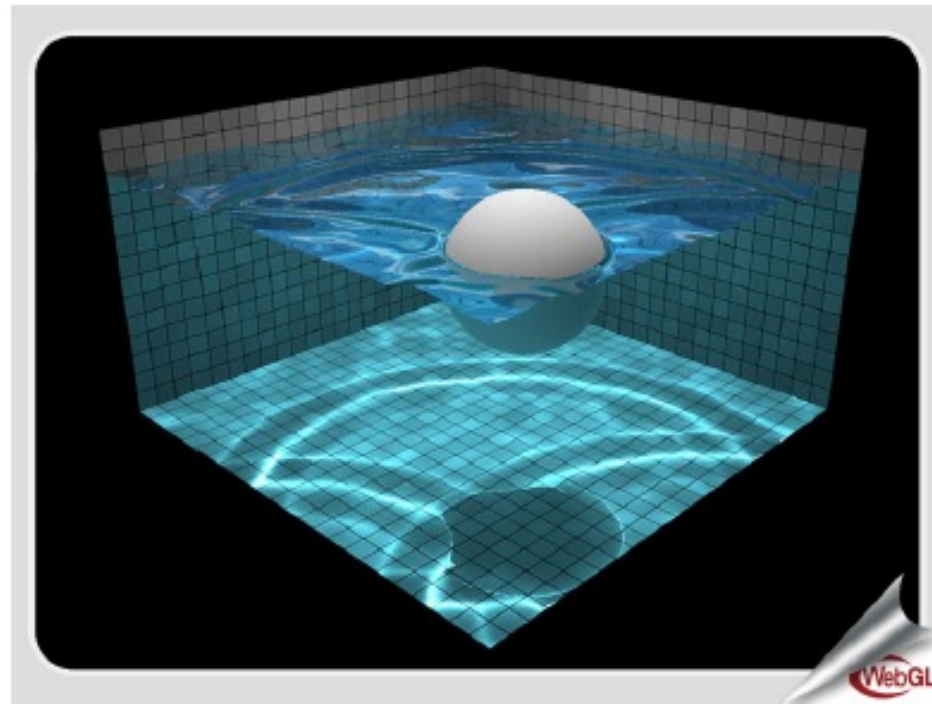
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```



Esempio WebGL

- Grafica 3D interattiva



<http://madebyevan.com/webgl-water/>

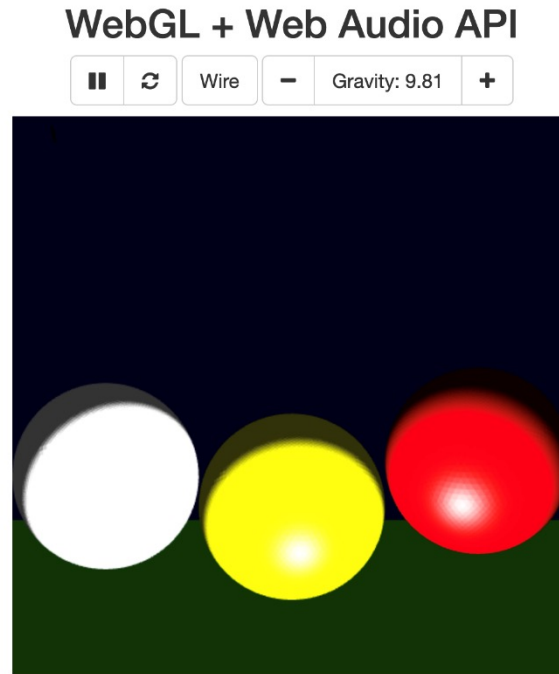
Esempio Web Audio

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <button id="play">Play</button>
    <script>
      // audio grafo
      const context = new AudioContext();
      // interfaccia
      var playButton = document.getElementById("play");
      playButton.addEventListener("click", PlayAudio);
      function PlayAudio() {
        // creazione nodi audio
        var oscillator = new OscillatorNode(context);
        var gainMaster = new GainNode(context);
        // connessioni
        oscillator.connect(gainMaster);
        gainMaster.connect(context.destination);
        // impostazione parametri (gain -30 dBFS)
        oscillator.type = "sine";
        oscillator.frequency.setValueAtTime(20, context.currentTime);
        oscillator.frequency.exponentialRampToValueAtTime(20000, context.currentTime+10);
        gainMaster.gain.value = 0.0316;
        // play audio per 10 secondi
        oscillator.start(context.currentTime);
        oscillator.stop(context.currentTime + 10);
      }
    </script>
  </body>
</html>
```



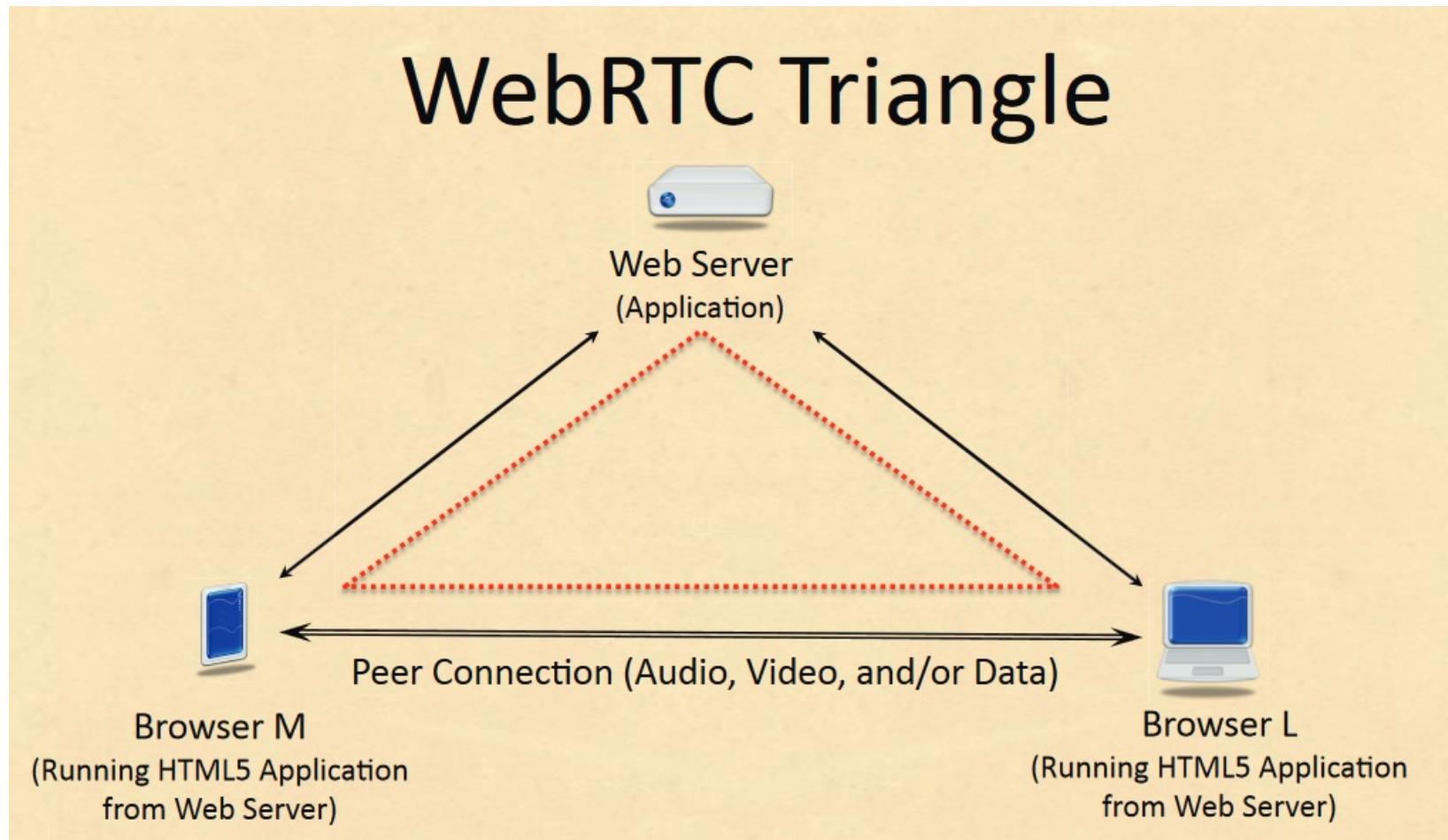
Sintesi digitale del suono: al click del pulsante si attiva un audio con un oscillatore sinusoidale

Esempio WebGL e Web Audio



<https://www.ime.usp.br/~fabiog/webaudio/webgl/index.html>

Esempio WebRTC



WEB APPLICATION FRAMEWORK

Cosa sono i framework?

- Un **framework** è un'architettura logica sulla quale un software può essere progettato e realizzato
- È definito come un'applicazione o un insieme di moduli per consentire lo **sviluppo agile di applicazioni** mediante l'utilizzo di *library* e/o funzionalità già rilasciate
- È uno strumento fondamentale perché facilita lo sviluppo da parte del programmatore di applicazioni web complesse
- Un **framework web** è uno strumento software che permette di realizzare applicazione web, consentendo di ottimizzare tempi, costi e benefici nella programmazione di un sito

Web Application Framework

- L'utilizzo dei **framework** permette al programmatore di utilizzare una serie di funzionalità senza andare a riscrivere ogni volta tutto da capo
- Il programmatore ha quindi a disposizione una serie di classi relative a funzionalità di uso comune come ad esempio
 - Autenticazione utenti
 - Gestione dei form e validazione dei dati
 - Gestione località e multilingual
 - Struttura delle directory e dei file
 - Interfacce utenti

Web Application Framework (2)

- Alcuni tra i più popolari Framework Web
 - **jQuery**: semplifica la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML
 - **React**: creazione di interfacce utente
 - **Angular**: sviluppo di applicazioni web
 - **ASP.NET**: sviluppo di applicazioni web e di servizi web
 - **Express**: progettato per creare web application e API
 - **Spring**: sviluppo di applicazioni su piattaforma Java

UN ESEMPIO DI FRAMEWORK: REACT

React

- **React** è un framework JavaScript open-source per creare **interfacce utente web**
- È stato sviluppato e rilasciato da Facebook nel 2013
- È ad oggi mantenuto e aggiornato da Meta (già Facebook), attualmente la versione è la V17.0.2 (August 2021)



<https://reactjs.org/>

React (2)

- Lo sviluppo di una pagina Web avviene attraverso la scrittura di cosiddetti componenti che manipolano il DOM per la creazione di elementi di interfaccia utente
- Invece di manipolare direttamente il DOM del browser, React crea un **DOM virtuale** in memoria, dove esegue tutte le manipolazioni necessarie, prima di apportare le modifiche al DOM del browser
- Permette allo sviluppatore di costruire interfacce complesse attraverso la composizione di semplici "mattoncini"

Componenti: funzioni e classi

- Esistono due tipi di componenti
 - Componenti di tipo **function**
 - Componenti di tipo **class**
- Entrambi i tipi di componenti sono obbligati a "restituire" codice HTML attraverso la keyword **return**

React e HTML

- Il modo più rapido per inserire React nei file HTML è aggiungere i link agli script del framework (utile per le fasi test)

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

    <div id="mydiv"></div>

    <script type="text/babel">
      function Hello() {
        return <h1>Hello World!</h1>;
      }

      ReactDOM.render(<Hello />, document.getElementById('mydiv'))
    </script>

  </body>
</html>
```



Hello World!

Babel permette di scrivere la sintassi JSX (estensione della sintassi JavaScript) e ES6 (versione 6 Javascript)

React e HTML (2)

- Per ambienti di produzione invece si installa il **React environment** sul server

```
import React from "react";  
import ReactDOM from "react-dom";
```

```
function Hello(props) {  
  return <h1>Hello World!</h1>;  
}
```

```
ReactDOM.render(<Hello />, document.getElementById("root"));
```



Hello World!

React Render HTML

- L'obiettivo di **React** è rendere l'HTML in una pagina web, utilizzando una funzione chiamata **ReactDOM.render()**
- Lo scopo della funzione è visualizzare il codice HTML specificato all'interno dell'elemento

```
import React from 'react';  
import ReactDOM from 'react-dom';
```

```
const myelement = (  
  <table>  
    <tr>  
      <th>Name</th>  
    </tr>  
    <tr>  
      <td>John</td>  
    </tr>  
    <tr>  
      <td>Elsa</td>  
    </tr>  
  </table>  
>);
```



| Name |
|------|
| John |
| Elsa |

```
ReactDOM.render(myelement, document.getElementById('root'));
```

Esempio

- Aggiungere un pulsante «like»

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
</head>
<body>
  <p>
    This is the first comment.
    <div class="like_button_container" data-commentid="1"></div>
  </p>
  <p>
    This is the second comment.
    <div class="like_button_container" data-commentid="2"></div>
  </p>
  <p>
    This is the third comment.
    <div class="like_button_container" data-commentid="3"></div>
  </p>
  <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
  <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
  <script src="like_button.js"></script>
</body>
</html>
```



This is the first comment.

Like

This is the second comment.

Like

This is the third comment.

Like

Esempio (2)

- like_button.js

```
const e = React.createElement;

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked: false };
  }

  render() {
    if (this.state.liked) {
      return 'You liked comment number ' + this.props.commentID;
    }

    return e(
      'button',
      { onClick: () => this.setState({ liked: true }) },
      'Like'
    );
  }
}

document.querySelectorAll('.like_button_container')
  .forEach(domContainer => {
    const commentID = parseInt(domContainer.dataset.commentid, 10);
    ReactDOM.render(
      e(LikeButton, { commentID: commentID }),
      domContainer
    );
  });
```



This is the first comment.

Like

This is the second comment.

You liked comment number 2

This is the third comment.

Like

React Tutorial

- <https://reactjs.org/tutorial/tutorial.html>
- <https://www.w3schools.com/react/>