Insegnamento di Tecnologie Web

CdS In Informatica

(A.A. 2022-23)

Esame scritto del 07/07/2023

Nome:

Cognome:

Matricola:

Corso di Studi

Anno di frequenza

Come specificato nel piano di studi: o "2022-23" " oppure "precedente".

Attenzione:

- Questi computer sono limitati ad accedere solo ad alcuni siti: `eol.unibo.it`, `virtuale.unibo.it`, `developer.mozilla.org`, `getbootstrap.com` e `site212248.tw.cs.unibo.it`. Non funzionano Google, stack overflow, etc.
- *Rispondete solo negli spazi delimitati dai blocchi* ``` `qui la risposta` ```*, senza modificarli o eliminarli.*
- *Consegnate* solo *questo file. Copiate ed incollate dentro agli appositi spazi la vostra risposta per intero.*
- *Potete decidere se inserire il CSS inline nel file HTML o metterlo in un file esterno. Nel secondo caso inserite l'elemento nella posizione corretta e mettete il CSS in un blocco separato.*
- *You can use either English or Italian for your answers.*
- *Per favore, per favore, per favore: nesun erore di ortografia. Cuesta è un universita e non la squola elementale.*

Domanda #1 - Domande di base (6 punti totali)

a) Semantic Web

Rappresentare in RDF i seguenti dati: “Angela Davis è nata a Birmingham, Alabama il 26 gennaio 1944. E’ autrice del libro “Donne, Razza e Classe”, del 1981.” Potete usare un formato a vostra scelta tra Turtle, XML-RDF o JSON-LD.

```
@prefix my: <...>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix anobii: <http://www.anobii.com/books/>.

db:angela_davis
  rdf:type foaf:person;
  foaf:name "Umberto Eco";
  foaf:born [
    foaf:place my:birningam;
    foaf:date '26 gennaio 1944'.
    ];
  foaf:made anobii:donne_razza_classe;
  foaf:work my:autrice.

my:birningam
  a foaf:city;
  foaf:state foaf:alabama.

anobii:donne_razza_classe
  a <http://purl.or/ontology/bibo/Book>;
  foaf:title "donne razza classe";
  foaf:relase_date "1981" .
```

b) Codifica caratteri

Quanti byte sono richiesti in UTF-8 per rappresentare le seguenti parole?

- Mongolo:
- Curdo: merheba
- Polacco:
- Portoghese: olá
- Italiano: ciao
- Afrikaans: hi

```
* Mongolo: ![](./images/ciao_mongolo.png){ height=14px } : 5 per le non americana 2 byte per
* Curdo: merheba : 7 byte
* Polacco: ![](./images/ciao_polacco.png){ height=10px } : 7 byte
```

```
* Portoghese: olá : 4 byte
* Italiano: ciao : 4 byte
* Afrikaans: hi : due byte
```

c) CSS

Avendo un'immagine con classe "rotating-circle", completare il seguente codice CSS per generare una animazione lineare infinita che giri su sé stessa. Per provare, si può usare l'immagine SVG posta in "circle.svg" e inserirla in un file HTML.

```
.rotating-circle {
    transform-origin: ____;
    animation: rotate-circle ____ ____ ;
}
@keyframes rotate-circle {
  0% {
    transform: _____________;
  }
  100% {
    transform: _____________;
  }
}
```

```
.rotating-circle {
    transform-origin: 50% 50%;
    animation: rotate-circle 3s linear infinite ;
}
@keyframes rotate-circle {
  0% {
    transform: rotate(0deg);
  }
  100% {
    transform: rotate(360deg);
  }
}
```

d) Javascript

Descrivere le IIFE in Javascript; fornire un esempio non banale e non presente nelle slide.

```
IIFE `e quando invociamo una funzione subito dopo averla devinita esempio

const valueOfciaoPlus2=(function (){
  return document.querySelector('#.ciao').value+2
})()
```

questo `e spesso utilizzato per esmpio per le funzioni che vogliamo eseguire subito qunado v
essendo che metterlo tutto nello scope globale il codice perde di espressione.

```
function main(){
  fetch('/api/getcontent', { method: "GET" })
  .then((response) => { if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`)
  .then((json) => {
    displayContent(json);
    }) .catch((error) => { throw new Error(error); });
}()
```

Domanda #2 - HTML + CSS (10 punti)

Scrivere il codice HTML e CSS cercando di riprodurre la seguente pagina web:
Logo e immagine a sinistra (logo.jpg, vesuvio.jpg) si trovano nello zip
scaricabile da EOL. E' possibile utilizzare Bootstrap.

Codice HTML *(ed eventualmente CSS interno)*

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
    <link rel="stylesheet" href="https://www.fabiovitali.it/TW/lib/bootstrap.min.css">
    <script src="https://www.fabiovitali.it/TW/lib/jquery-3.6.0.js"></script>
    <script src="https://www.fabiovitali.it/TW/lib/bootstrap.js"></script>
    <style>
.nav-link{
  color:gray;
}
    </style>
  </head>

  <body>
    <nav class="navbar bg-body-tertiary">
      <div class="container justify-content-end">
        <ul class="nav" style='color:gray' >
          <li class="nav-item">
            <a class="nav-link" >Love</a>
          </li>
          <li class="nav-item">
            <a class="nav-link ">Shop</a>
          </li>
        </ul>
      </div>
    </nav>
    <div class='container ' >
      <div class='d-flex my-5 justify-content-center' >
        <img src='logo.jpg' width='200px'/>
      </div>
        <div class="row">
          <div class="col w--100">
            <img src='vesuvio.jpg' class='w-100' />

          </div>
      <div class="col  justify-content-center" style="height: auto; display: flex; flex-dire
```

```
        <h1>
          NG RECORDS
        </h1>
        <p>
        NG records  is a labl run from vesuvio.
        </p>
        <p>
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        NG records  is a labl run from vesuvio.
        </p>
        <p>
        NG records  is a labl run from <a href="#">vesuvio.</a>
        </p>
      </div>
    </div>
  </body>
</html>
```

Codice CSS ***(solo se esterno)***

Domanda #3 - Javascript (12 punti)

Creare un'interfaccia per interagire con un'API esistente, Dummy Image (`https://dummyimage.com`). Partendo dall'URL di base, bisogna creare una serie di input che aggiungono parametri all'URL per poter giungere al risultato. E' possibile utilizzare un framework se ritenuto necessario.

NOTA BENE: qualora si consegnino tutte le parti insieme, inserire un commento per differenziarle e.g. `"//p.2"`

Parte 1 (2 punti) Scrivere il codice HTML (se si vuole, anche CSS) e JS per modificare l'URL dell'API per poter manipolare l'URI e modificare i parametri.

- Grandezza: e.g. `https://dummyimage.com/600x400/`. L'input deve essere numerico.
- Colore sfondo: `https://dummyimage.com/600x400/color/`. L'input deve poter inserire solamente colori. Nota bene: l'API non accetta il #.
- Colore testo: `https://dummyimage.com/600x400/color/textColor/`
- Formato: `https://dummyimage.com/600x400/color/textColor.format`
- Testo: `https://dummyimage.com/600x400/000/fff.png&text=LoremIpsum` Un bottone alla fine fa console.log dell'URL sulla console

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
    <link rel="stylesheet" href="https://www.fabiovitali.it/TW/lib/bootstrap.min.css">
    <script src="https://www.fabiovitali.it/TW/lib/jquery-3.6.0.js"></script>
    <script src="https://www.fabiovitali.it/TW/lib/bootstrap.js"></script>
  </head>

  <body>
    <form>
      <label> height</label>
      <input type='number' id='w'/>
      <label> width</label>
      <input type='number' id='h'/>
      <label> color</label>
      <input type='text' id='c'/>
      <label> text color</label>
      <input type='text' id='ct'/>
      <label> formato testo</label>
      <input type='text' id='ft'/>
```

```
      <label> testo</label>
      <input type='text' id='t'/>
      <button id='createUrl' type='button'>
        print url
      </button>

    </form>
    <script src='es3.js' defer></script>
  </body>
</html>

------es3.js

function createUrl(ev){
  ev.preventDefault();

  const w=document.querySelector('#w').value;
  const h=document.querySelector('#h').value;
  const c=document.querySelector('#c').value;
  const ct=document.querySelector('#ct').value;
  const ft=document.querySelector('#ft').value;
  const t=document.querySelector('#t').value;
  console.log(`https://dummyimage.com/${w}x${h}/${c}/${ct}.${ft}&text=${t}`);
}

document.querySelector('#createUrl').onclick=createUrl;
```

Parte II (4 punti) Partendo dal codice dell'esercizio precedente, togliere l'istruzione console.log dell'URL sulla console e fare in modo che l'immagine venga direttamente mostrata dentro un box sotto gli input e il bottone.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
    <link rel="stylesheet" href="https://www.fabiovitali.it/TW/lib/bootstrap.min.css">
    <script src="https://www.fabiovitali.it/TW/lib/jquery-3.6.0.js"></script>
    <script src="https://www.fabiovitali.it/TW/lib/bootstrap.js"></script>
  </head>

  <body class='container'>
    <form class=' d-flex flex-column' >
      <label> height</label>
```

```
      <input type='number' id='w'/>

      <label> width</label>
      <input type='number' id='h'/>
      <label> color</label>
      <input type='text' id='c'/>
      <label> text color</label>
      <input type='text' id='ct'/>
      <label> formato testo</label>
      <input type='text' id='ft'/>
      <label> testo</label>
      <input type='text' id='t'/>
      <button id='createUrl' type='button'>
        show img
      </button>

    </form>
    <div id='main'>
    </div>
    <script src='es3.js' defer></script>
  </body>
</html>

----- es3.js

function createUrl(ev){
  ev.preventDefault();

  const w=document.querySelector('#w').value;
  const h=document.querySelector('#h').value;
  const c=document.querySelector('#c').value;
  const ct=document.querySelector('#ct').value;
  const ft=document.querySelector('#ft').value;
  const t=document.querySelector('#t').value;
  const url=(`https://dummyimage.com/${w}x${h}/${c}/${ct}.${ft}&text=${t}`);

  document.querySelector('#main').innerHTML=`<img src='${url}'/>`
}


document.querySelector('#createUrl').onclick=createUrl;
```

Parte III (6 punti) Partendo dal codice dell'esercizio precedente, aggiungere un bottone che fa una chiamata all'API con ogni parametro

generato casualmente.

- Per generare le dimensioni casualmente, usare `Math.random()`.
- Per i colori (solo esadecimali): `const randomHexColor = Math.floor(Math.random()*16777215).toString(16);`
- Per il testo: usare sempre `Math.random()` per generare una stringa di testo di max 6 caratteri casuali basandosi sull'alfabeto. Altrimenti, usare una parola a caso dalla frase: "Lorem Ipsum Dolor Sit Amet".

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello world</title>
    <link rel="stylesheet" href="https://www.fabiovitali.it/TW/lib/bootstrap.min.css">
    <script src="https://www.fabiovitali.it/TW/lib/jquery-3.6.0.js"></script>
    <script src="https://www.fabiovitali.it/TW/lib/bootstrap.js"></script>
  </head>

  <body class='container'>
    <form class=' d-flex flex-column' >
      <label> height</label>
      <input type='number' id='w'/>

      <label> width</label>
      <input type='number' id='h'/>
      <label> color</label>
      <input type='text' id='c'/>
      <label> text color</label>
      <input type='text' id='ct'/>
      <label> formato testo</label>
      <input type='text' id='ft'/>
      <label> testo</label>
      <input type='text' id='t'/>
      <button id='createUrl' type='button'>
        show img
      </button>
      <button id='randomUrl' type='button'>
        random
      </button>

    </form>
    <div id='main'>
    </div>
    <script src='es3.js' defer></script>
  </body>
```

```
</html>

---- es3.js
function createUrl(ev){
  ev.preventDefault();

  const w=document.querySelector('#w').value;
  const h=document.querySelector('#h').value;
  const c=document.querySelector('#c').value;
  const ct=document.querySelector('#ct').value;
  const ft=document.querySelector('#ft').value;
  const t=document.querySelector('#t').value;
  createUrlAndDisplay(w,h,c,ct,ft,t);
}

function createUrlAndDisplay(w,h,c,ct,ft,t){
  const url=(`https://dummyimage.com/${w}x${h}/${c}/${ct}.${ft}&text=${t}`);
  document.querySelector('#main').innerHTML=`<img src='${url}'/>`
  console.log(url)
}

function randomUrl(ev){
  ev.preventDefault();

  const rHc =()=>Math.floor(Math.random()*16777215).toString(16)
  const rI=(i)=>parseInt(Math.random()*i)
  const w=rI(1000)
  const h=rI(1000)
  const c= rHc()
  const ct=rHc()
  const ft='jpg'
  const words='Lorem Ipsum Dolor Sit Amet'.split(' ')
  const t=words[rI(words.length)]

  createUrlAndDisplay(w,h,c,ct,ft,t);
}


document.querySelector('#createUrl').onclick=createUrl;

document.querySelector('#randomUrl').onclick=randomUrl;
```

Domanda #4 - Framework (6 punti)

Scrivere il codice necessario per un componente funga da galleria di immagini.

- Si parta da un framework a vostra scelta, e.g. React, Angular, Vue, Svelte, etc.
- Si può suddividere l'esercizio in più componenti, ma devono essere tutti richiamati nel componente padre.
- Il componente riceve un array di oggetti immagine di partenza; ogni oggetto è composto da URL dell'immagine, titolo e descrizione.
- La galleria mostra un'immagine alla volta. Oltre all'immagine, titolo e descrizione, ci deve essere un modo per navigare tra un'immagine e l'altra.
- Puoi usare questo screenshot per immaginare il risultato:

```
// props ha il fild imgs che conteiene un array di immagini con {src:'',title:'',descrizione
funciton App(props){
  // prendiamo solo l'src
  cosnt immagini=props.imgs.map((c)=>c.src);
  const [img,setImg]=React.useState(0);
  const changeImg=(value){
    // se va fuori dagli estremi mette aposto
    if(value>=imgs.length){
      value=0;
    }else if(value<=-1){
      value=immagini.length-1;
    }
    // cambia lo stato
    setImg(vlaue);
  }

  return <>
    <h1>Titolo<h1>
    <p> lorem etc...</p>
    <img src={immagini[img]} title={props.imgs[img].title} alt={props.imgs[img].descrizione]
    <button class='sidebutton-left' onClick={()=>changeImg(img+1)} >next</button>
    <button class='sidebutton-right' onClick={()=>changeImg(img-1)} >prev</button>
    {immagini.map((value,index)=>{
      return <img key={index} src={value} onClick={()=>changeImg(index)} title={props.imgs[i
    })}

  </>;
}
```