Insegnamento di Tecnologie Web

CdS In Informatica

(A.A. 2021-22)

Esame scritto del del 23/06/2022

Prova in presenza

Questa soluzione è stata effettivamente consegnata il giorno dello scritto e successivamente valutata. Si noti che sono stati aggiunti due trentesimi bonus a ogni studente, per cui il punteggio massimo raggiungibile è 35/30 anziché i 33/30 suggeriti dalla somma dei singoli esercizi. Questo è stato il primo appello in cui i professori, sollecitati dagli studenti, si sono degnati di specificare il peso di ogni esercizio sulla prova. Per il resto, come al solito, gli esercizi sono di una lunghezza totale sproporzionata rispetto alle due ore concesse.

Nome:

Cognome:

Matricola:

Corso di Studi

Solo se diverso da Informatica Triennale

Anno di frequenza

Come specificato nel piano di studi: o "2021-22" " oppure "precedente".

Attenzione:

- Questi computer sono limitati ad accedere solo ad alcuni siti: `eol.unibo.it`, `virtuale.unibo.it`, `developer.mozilla.org` e `getbootstrap.com`. Non funzionano Google, stack overflow, W3 Schools, 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 (4 punti)

> Queste risposte hanno ottenuto 1.00/1.00 (successivamente moltiplicato per i 4 punti dell'esercizio), nonostante nel punto d la parola richiesta non abbia almeno cinque lettere come richiesto dal testo.

a) HTTP

"Un metodo HTTP è sicuro (*safe*) se lo si usa all'interno di una connessione HTTPS" Questa affermazione è vera o falsa? Perché?

```
Questa affermazione e' falsa:  esistono metodi HTTP non sicuri ma comunque usabili in connes
metodi sicuri usabili in connessioni non HTTPS. Nel caso dei metodi HTTP, la "sicurezza" nor
alla crittografia, bensi' alla garanzia che l'invocazione del metodo lasci inalterato lo sta
server.
```

b) HTML

Cosa sono gli elementi generici in HTML?

```
In HTML, gli elementi generici (vedi div e span) sono elementi ai quali non sono state assoc
caratteristiche di alcun tipo (ne' stilistico ne' semantico), fatta eccezione per l'essere e
blocco (div) o in linea (span). Il loro ruolo e' quello di elementi "fantoccio" quando vogli
specificare caratteristiche stilistiche o semantiche solamente usando attributi.
```

c) CSS

Dato il fragmento `<p class="saluto">Hello world</p>` e le regole CSS:

```
[saluto] {color:red;}
#saluto {color:blue;}
saluto {color:green;}
```

, In che colore verranno scritte le parole del paragrafo?

```
Le parole del paragrafo non sono vincolate da nessuna delle tre regole:
- manca un attributo saluto (saluto e' il valore)
- manca un id=saluto
- l'elemento non e' di tag saluto
Quindi, salvo stili  ereditati da elementi genitori, viene scelto il colore predefinito per
```

d) Codifica caratteri

Si faccia un esempio di una parola dell'italiano comune di almeno cinque lettere che abbia una lunghezza diversa in byte a seconda che venga condificata in ISO Latin 1, in UCS-2 o in UTF-8. Si specifichino anche le rispettive lunghezze.

```
dà
```

```
In UTF8 ha 1 + 2 = 3 byte
In LATIN-1 ha 1 + 1 = 2 byte
In UCS-2 ha ha 2 + 2 = 4 byte
```

Oppure, proposta di soluzione con una parola di almeno 5 lettere:

```
città
In UTF8 ha 1+1+1+1+2 = 6 byte
In LATIN-1 ha 1+1+1+1+1 = 5 byte
In UCS-2 ha ha 2+2+2+2+2 = 10 byte
```

domanda HTML e CSS

Figure 1: domanda HTML e CSS

Domanda #2 - HTML + CSS (10 punti)

Scrivere il codice HTML e CSS (bootstrap è ammesso solo se importato correttamente nella pagina) della seguente pagina web. Le immagini sono fornite in uno zip scaricabile da EOL. Il codice deve funzionare su Firefox. Può essere usato come base il documento `base.html`. Non è importante essere totalmente precisi con colori e misure, ma essere ragionevolmente attenti alle differenze tra elemento ed elemento. Se il file non si apre correttamente nel browser, il punteggio è 0.

> Questo esercizio ha ottenuto 0.80/1.00, poi moltiplicato per 10 punti. Non so esattamente cosa, oltre al pie' di pagina parzialmente sacrificato per questioni di tempo, Sovrano avrebbe voluto più preciso.

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>
</head>

<body>
  <nav class="navbar navbar-expand-lg pb-5">
    <div class="container-fluid justify-content-between">
      <a class="navbar-brand link-dark" href="#">
        <img
          src="img/img2/logo2.png"
          alt="" height="64" />
      </a>
      <div id="navbarNav" class="container-fluid">
        <div class="row">
          <div class="col-12 d-flex flex-row-reverse">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link text-dark" href="#">
              <img src="icon.png" height="24" alt="" />Contact Sales</a>
```

```
        </li>
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Support</a>
        </li>
        <li class="nav-item">
            <button type="button" class="btn btn-primary">
              @</button>
        </li>
      </ul>
        </div>
      </div>
      <div class="row">
        <div class="col-12 d-flex flex-row-reverse">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Products</a>
        </li>
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Solutions</a>
        </li>
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Resources</a>
        </li>
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Research</a>
        </li>
        <li class="nav-item">
          <a class="nav-link text-dark" href="#">
            <img src="icon.png" height="24" alt="" />Company</a>
        </li>
      </ul>
        </div>
      </div>
    </div>
  </div>
</nav>
<div class="container-flex">
  <div class="row">
    <div class="col-12">
      <div class="card bg-dark text-white">
        <img src="img/img2/home.png" class="card-img" alt="" >
```

```
        <div class="card-img-overlay p-5 m-5">
          <h1 class="card-title">Changing your idea of what robots can do.</h1>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-12 text-center text-light bg-primary">
    ON-DEMAND: Maximize Site Efficiency with Spot's Newest Capabilities
    <button type"button" class="btn btn-primary border">WATCH NOW</button>
  </div>
</div>
<div class="row bg-dark">
  <div class="col-6">
    <div class="card bg-dark text-white">
      <img src="img/img2/spot-banner-home.jpg" class="card-img" alt="" >
        <div class="card-img-overlay">
          <h1 class="card-title">Spot</h1>
          <p class="card-text"> Automate data capture and inspection</p>
          <button type="btn" class="btn border btn-white text-light">SEE SPOT</button>
        </div>
    </div>
  </div>
  <div class="col-6">
    <div class="card bg-dark text-white">
      <img src="img/img2/stretch-banner-home.jpg" class="card-img" alt="" >
        <div class="card-img-overlay">
          <h1 class="card-title">Stretch</h1>
          <p class="card-text">Mobile, automated case handling for more efficient wareho
          <button type="btn" class="btn border btn-white text-light">SEE STRETCH</buttor
        </div>
    </div>
  </div>
</div>
</div>

      </div>
    </div>
    <div class="row d-flex align-items-stretch">
      <div class="col-6 p-2">
        <div class="card align-self-stretch">
          <div class="card-body">
            <h5 class="card-title">Spot Inspection Solution</h5>
            <p class="card-text text-primary">Learn more</p>
          </div>
```

```
            <img src="img/img2/spot-inspection.png" alt="" />
          </div>
        </div>
        <div class="col-6 p-2">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Stretch Warehouse Solution</h5>
              <p class="card-text text-primary">Learn more</p>
            </div>
            <img src="img/img2/stretch-warehouse.png" alt="" />
          </div>
        </div>
      </div>
<div class="row p-2">
  <div class="col-12 p-2">
    <div class="card bg-dark text-white">
      <img src="img/img2/cta-atlas.jpg" class="card-img" alt="" >
        <div class="card-img-overlay p-5 m-5">
          <h1 class="card-title">Atlas</h1>
          <p class="card-text">Discover the world's most advanced humanoid robot.</p>
          <button type="button" class="btn btn-light border">LEARN MORE</button>
        </div>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col-1 bg-primary align-middle">
  </div>
  <div class="col-3 bg-primary align-middle">
    <img class "text-center align-middle" src="img/img2/logo.png" alt="" height="60"/>
  </div>
  <div class="col-2 bg-dark text-white">
    <b>PRODUCTS</b> <br>
    SPOT <br>
    SPOT ARM <br>
  </div>
  <div class="col-2 bg-dark text-white">
    <b>SOLUTIONS</b> <br>
    INSPECTION <br>
    ASSET MANAGEMENT <br>
  </div>
  <div class="col-2 bg-dark text-white">
    <b>RESOURCES</b> <br>
    CASE STUDIES <br>
    WEBINARS <br>
```

```
      </div>
      <div class="col-2 bg-dark text-white">
        <b>COMPANY</b> <br>
        ABOUT <br>
        CAREERS <br>
      </div>
    </div>
  </div>
</body>

</html>
```

Codice CSS *(solo se esterno)*

```
# Non ho usato CSS.
```

Domanda #3 - Javascript (10 punti)

Si consideri un sito di un organizzazione di biglietteria per eventi (concerti, spettacoli, eventi sportivi, sagre, ecc.). Alcuni eventi sono individuali, altri appartengono ad una sequenza di eventi individuali equivalenti ma in date e/o luoghi diversi (ad esempio, una tournee dello stesso spettacolo in città diverse, o una sagra che si svolge su più giorni. In questo caso l'utente prima sceglie l'evento, e poi, sulla base della disponibilità, la singola data/luogo. Alcuni eventi hanno un biglietto generico, altri sono collegati ad uno specifico posto a sedere in una griglia predefinita. Per semplicità assumiamo che ogni location sia descritta da una griglia rettangolare di `n x m` posizioni. Ogni posizione può essere associata ad uno e uno solo dei seguenti valori:

- palco (non vendibile)
- non-posto (corridoi, colonne, spazi liberi di ogni tipo)
- posto riservato
- posto già venduto
- posto libero riservato a carrozzine e disabili e in vendita a prezzo 1
- posto libero in vendita a prezzo 2
- posto libero in vendita a prezzo 3
- posto libero in vendita a prezzo 4

Usate il documento allegato, `base.html`, per fare le vostre prove. Il file contiene, già configurati, un accesso a jquery e a bootstrap. Potete usare i servizi `http://site202100.tw.cs.unibo.it/info` oppure `http://www.fabiovitali.it/TW/test/2021/doResponse.php` per effettuare prove e debugging. Non dovete descrivere l'API, solo ipotizzarla con chiarezza e usarla in maniera corretta. Basandosi, dove si ritiene, su uno o più framework Javascript a piacere tra quelli illustrati a lezione:

Il sistema espone quattro servizi:

- `https://www.site.com/listing?from=YYYY-MM-DD&to=YYYY-MM-DD` fornisce un JSON con tutti gli eventi tra le due date specificate. Ogni evento ha un id associato, uno o più campi descrittivi (nome artista, nome spettacolo, cast, etc.). Non specifica né location né date (a parte nella sezione descrittiva).
- `https://www.site.com/show?id=XXX` fornisce l'elenco di date, location dello show XXX (dove XXX è l'id definito precedentemente). Ogni combinazione di show, data e location è associato ad un eventId diverso.
- `https://www.site.com/event?id=XXX` fornisce la mappa dei posti disponibili per l'evento XXX (dove XXX è l'eventId definito precedentemente). Ogni mappa riporta tutti i biglietti disponibili e quelli già venduti per il singolo evento, e i posti liberi (definiti con un placeID) vengono visualizzati sotto forma di elemento cliccabile.
- `https://www.site.com/buy?place=XXX` permette all'utente di acquistare il biglietto relativo al posto in questione.

> Questo esercizio è stato valutato 1.00/1.00 (poi moltiplicato per 10 punti) perché a Vitali importa solo che ci sia l'"idea". Si noti che infatti la consegna è interpretata in modo molto lassista e a tratti neanche seguita. Non ci si è neanche preoccupati di produrre codice funzionante.

Parte I ***(Solo HTML oppure HTML + Javascript)*** Si scriva il codice HTML che contiene la struttura della pagina della applicazione, con la possibilità di scegliere progressivamente show, evento, e posto (si usino in maniera corretta ed appropriata immagini, descrizione, prezzo e giorni disponibili),

```
<html>

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

<body>
  <h1>Biglietteria</h1>
  <h2>Scegli il tuo spettacolo:<h2>
  <div id="shows">
  </div>
  <h2>Scegli il tuo evento:<h2>
  <div id="events">
  </div>
  <h2>Scegli il tuo posto:<h2>
  <div id="seats">
  </div>
  <form>
    <input type="submit" value="Prenota" id="prenotation"/>
  </form>
</body>

</html>
```

```
function populateShows(shows) {
  for (let i = 0; i < shows.length; ++i)
    document.getElementById("shows").innerHTML +=
      `<article>
        Artista: <span id="artist${i}">${shows[i].artist}</span><br />
        Spettacolo: <span id="name${i}">${shows[i].name}</span><br />
        Cast: <span id="cast${i}">${shows[i].cast}</span><br />
       </article>`;
}
```

```
function populateEvents(events) {
  for (let i = 0; i < events.length; ++i)
    document.getElementById("events").innerHTML +=
      `<article>
        Data: <span id="date${i}">${events[i].date}</span><br />
        Luogo: <span id="location${i}">${events[i].location}</span><br />
       </article>`;
}

function populateSeats(seats) {
    var body = document.getElementById('seats');
    var tbl = document.createElement('table');
    var tbdy = document.createElement('tbody');
    // head
    var tr = document.createElement('tr');
    for (var cell of Object.keys(val[0])) {
        var td = document.createElement('th');
        td.appendChild(document.createTextNode(cell))
        tr.appendChild(td)
    }
    tbdy.appendChild(tr);
    // body
    for (var row of val) {
        var tr = document.createElement('tr');
        for (var cell of Object.values(row)) {
                var td = document.createElement('td');
                td.innerhtml= `<button id="seat-${i}-${j}">${cell}</button>`;
                tr.appendChild(td)
        }
        tbdy.appendChild(tr);
    }
    tbl.appendChild(tbdy);
    body.appendChild(tbl)
}
```

Parte II ***(Riempire con solo Javascript solo se la parte I contiene solo HTML)*** Si scrivano uno o più script Javascript che popolino progressivamente la pagina delle informazioni scelte dall'utemte, prima selezionando gli eventi dispnibili in un certo intervallo di date, poi scegliendo uno show, poi scegliendo una data e una location, e infine scegliendo il posto.

```
function getShows(from, to) {
  fetch(
    `https://www.site.com/listing?from=${from}&to=${to}`,
    { method: "GET" }
  )
```

```
    .then((response) => {
          if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`);
          return response.json();
      }
    })
    .then((json) => {
        populateShows(json);
      }
    })
    .catch((error) => {
      throw new Error(error);
    });
}

function getEvents(show) {
  fetch(
    `https://www.site.com/show?id=${show}`,
    { method: "GET" }
  )
    .then((response) => {
          if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`);
          return response.json();
      }
    })
    .then((json) => {
        populateEvents(json);
      }
    })
    .catch((error) => {
      throw new Error(error);
    });
}

function getSeats(event) {
  fetch(
    `https://www.site.com/event?id=${event}`,
    { method: "GET" }
  )
    .then((response) => {
          if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`);
          return response.json();
      }
    })
    .then((json) => {
        populateSeats(json);
      }
```

```
    })
    .catch((error) => {
      throw new Error(error);
    });
}

function singleBuy(seat) {
  fetch(
    `https://www.site.com/buy?place=${seat}`,
    { method: "POST" }
  )
    .then((response) => {
          if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`);
          return response.json();
      }
    })
    .then((json) => {
        alert("Biglietto acquistato con successo.");
      }
    })
    .catch((error) => {
      throw new Error(error);
    });
}
```

Parte III *(HTML + Javascript)* Si modifichino l'HTML e gli script in modo che sia possibile acquistare più biglietti in un'unica azione, senza necessariamente pagarli separatamente. I biglietti, tuttavia, sono sempre dello stesso evento nella stessa location e nella stessa data, ma non sono necessariamente contigui di posizione. Il sistema calcola il totale del costo del biglietto prima di eseguire l'acquisto. Si documenti il cambiamento dell'uri dell'API specificata sopra in caso di acquisti multipli. In questa sezione si presentino una copia di HTML e script presentati precedentemente e si applichino le modifiche necessarie per questa aggiunta.

```
function alertBeforeBuy(seats) {
    let price = 0;
    for (int i = 0; i < seats.length; ++i)
        price += seats[i].price;
    alert("Totale: $" + price);
}

// Cambiamento dell'URI: scelgo di usare
// www.site.com/multipleBuy?places=${seats},
// dove ora spedisco una sequenza ordinata e finita
// di identificativi di posti anziche' un posto unico.
```

```
function multipleBuy(seats) {
  fetch(
    `https://www.site.com/multipleBuy?places=${seats}`,
    { method: "POST" }
  )
    .then((response) => {
          if (!response.ok) throw new Error(`Errore HTTP ${response.status}.`);
          return response.json();
      }
    })
    .then((json) => {
        alert("Biglietti acquistati con successo.");
      }
    })
    .catch((error) => {
      throw new Error(error);
    });
}
```

Domanda #4 - Semantic Web (4 punti)

Scrivere in Turtle il grafo RDF della seguente frase, poi specificare quante triple contiene: «R. Madhavan, pseudonimo di Madhavan Balaji Ranganathan (Jamshedpur, 1º giugno 1970), è un attore e produttore cinematografico indiano.».

> Questa soluzione ha ottenuto 0.95/1.00, poi moltiplicato per i 4 punti dell'esercizio. Vado molto a caso negli esercizi di semantica, specialmente se corretti da Sovrano.

```
@prefix my: <...>.

my:madhavan
  my:nome "Madhavan Balaji Ranganathan";
  my:pseudonimo "R. Madhavan";
  my:citta-natale my:jamshedpure;
  my:data-di-nascita my:1970-06-01;
  my:carriera [
    my:tipo list;
    my:tipo_elementi: my:lavoro;
    my:primo my:attore;
    my:secondo my:produttore_cinematografico
  ].

my:jamshedpure
  my:tipo citta;
  my:nome "Jamshedpure";
  my:nazione my:india.

my:india
  my:tipo nazione;
  my:nome "India".

# Per come io ho scelto di modellizzare questa frase, essa contiene
# 14 triple.
```

Domanda #5 - Domanda di accessibilità (5 punti)

Il form contatti del sito `https://includeretutti.it` contiene diversi campi, tra cui:

- due campi di testo per inserire nome e cognome;
- un campo per scegliere il genere fra le tre opzioni “maschile”, “femminile” e “altro”, da implementare mediante pulsanti radio;
- una textarea per inserire il messaggio;
- una checkbox per indicare di aver letto ed accettare l’informativa sulla privacy;
- il pulsante per inviare il form.

Scrivere il codice HTML della pagina del form contatti rispettando tutti i criteri di accessibilità. Inserire anche i landmark ed i livelli di intestazione necessari.

> Questa soluzione ha preso 0.70/1.00, poi moltiplicati per i 5 punti dell’esercizio. Non ho avuto tempo di pensare né ai *landmark* né ai livelli di intestazione necessari.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Contattaci | tuttoaccessibile</title>
</head>

<body>
  [&ellip;]
  <form action="/invia/">
    <label for="name">Nome:</label>
    <input type="text" id="name" required>
    <label for="lastname">Cognome:</label>
    <input type="text" name="lastname" id="lastname" required>
    <fieldset>
            <legend>Genere</legend>
            <input id="radio1" type="radio">
            <label for="radio1">Maschile</label>
            <input id="radio2" type="radio">
            <label for="radio2">Femminile</label>
            <input id="radio1" type="radio">
            <label for="radio3">Altro</label>
          </fieldset>
    <label for="body">Testo del messaggio:</label>
```

```
    <textarea name="body" id="body" cols="30" rows="10"></textarea>
        <input type="checkbox" id="subscribeNews" name="subscribe" value="newsletter">
    <label for="subscribeNews">Ho letto e accetto l'informativa sulla privacy</label>
    <button type="submit">Invia messaggio!</button>
  </form>
  [&ellip;]
</body>

</html>
```