Rappresentazione dei dati

È necessario definire un formalismo flessibile per:

- Avere una separazione tra **contenuto**, **presentazione** e **navigazione**;
- Avere una definizione di *domini* o *contesti*;
- Avere una indipendenza dalla piattaforma (media) e supporto multilingue.

Questo formalismo è definito dai linguaggio di marcatura dei documenti.
Esistono due tipi di marcatura:

- **Marcatura procedurale**: descrive come processare il documento. Etichettano semplicemente parti del testo, disaccoppiando la struttura dalla presentazione del testo stesso.
  Esempi: PostScript, PDF, RTF, formato MS Word.
- **Marcatura descrittiva**: descrive la struttura logica del documento. Definiscono istruzioni per programmi che elaborino il testo al quale sono associate.
  Esempi: HTML, SGML, XML.
- **Marcatura di presentazione**: definiscono come visualizzare il testo al quale sono associate.

9.1 Markup language

Il linguaggio di markup è un insieme di regole sintattiche che definiscono la struttura e la presentazione di un documento (ad esempio un sito web). Vengono utilizzati per descrivere il contenuto e l'aspetto di un documento in modo che possa essere interpretato da un'applicazione software.

Esempi: TeX (e LaTeX), SGML, HTML, XHTML, XML

9.1.1 XML

XML , eXtensible Markup Language, si è imposto come standard de facto per lo scambio di *informazioni semi-strutturate*.

Caratteristiche

- Marcatura descrittiva e non procedurale
- Marcatura definisce la struttura logica del documento
- **Flessibilità**: le etichette possono cambiare in base all'applicazione
- Il tipo di documento specifica la struttura della marcatura ammessa
  Document Type Declaration o *DTD* è parte dello standard XML

Un documento che rispetta la specifica di XML è detto **well-formed** se:

- È costituito da un insieme di elementi annidati;
- Un elemento è costituito da una coppia di etichette di apertura e di chiusura e da quello che racchiudono.

```
<?xml version="1.0"?>
<media>
        <cds>
                <cd quantity="10">
                        <title>Glory and Consequence</title>
                        <artist>Ben Harper</artist>
                        <album>The Will to Live</album>
                        <album>Hello</album>
```

```
                </cd>
                <cd quantity="6">
                        <title>Bigmouth Strikes Again</title>
                        <artist>The Smiths</artist>
                        <album>Meat is Murder</album>
                </cd>
        </cds>
</media>
```

9.1.1.1 DTD - Document Type Declaration

> **Definizione di DTD**
>
> **DTD**, Document Type Declaration, è una *forma di linguaggio markup* utilizzato principalmente con XML per definire la struttura e il tipo di contenuto di un documento XML. Una DTD specifica quali elementi possono apparire in un documento XML, in che ordine possono apparire e quali attributi possono essere associati a ciascun elemento.

Un documento XML ha una **struttura arbitraria** quando:

- Ha elementi con nome arbitrario
- Ha attributi arbitrari in qualunque elemento
- Ha elementi organizzati in modo arbitrario

Un DTD specifica quali sono le strutture ammesse: specifica il **tipo del documento** e contiene **dichiarazioni dei tipi** di *elemento*, di *attributi* e di *entità*.

```
<?xml version="1.0"?>
<!DOCTYPE Media SYSTEM "media.dtd">
<Media>
...
</Media>

File media.dtd
<!ELEMENT media (cds | books)>
<!ELEMENT cds (cd*)>
<!ELEMENT cd (title, artist, album+)>
<!ATTLIST cd quantity CDATA #REQUIRED>
<!ELEMENT books (book*)>
<!ELEMENT book (title, author)>
<!ATTLIST book quantity CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

Limitazioni
DTD descrive solo la grammatica del file XML, non la struttura in modo dettagliato e/o i tipi. Questo porta a delle limitazioni, come per esempio:

- Restrizioni sul tipo di valore di un elemento o di un attributo
  Esempio: non si può indicare che il valore di un elemento o di un attributo non può essere un numero negativo
- Restrizioni di co-occorrenza
  Esempio: l'elemento "unità" dovrebbe essere presente solo quando è presente anche l'elemento "quantità"
- Flessibilità
  Esempio: l'elemento "commento" dovrebbe poter apparire dovunque

- Riutilizzo delle definizioni
  Esempio: non esiste il concetto di sottotipo/ereditarietà
- Integrità referenziale
  Esempio: non esistono le chiavi composte

9.1.1.2 XML Schema

Tecnica più sofisticata che risolve le limitazioni dei DTD.
Supporta le restrizioni sul tipo dei valori, tipi complessi e molti altri aspetti come integrità referenziale, eredità etc.

XML Schema è già specificato nella sintassi di XML, invece dei DTD. Inoltre è anche integrato nel namespace.
Tuttavia, è molto più complesso dei DTD.

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://w3.org/2001/XMLSchema">
        <xsd:element name="book" type="BookType"/>
                <xsd:complexType name="BookType">
                        <xsd:sequence>
                                <xsd:element name="title" type="xsd:string"/>
                                <xsd:element name="author"
                                                type="PersonType"
                                                minOccurs="1"
                                                maxOccurs="unbounded"/>
                                <xsd:complexType name="PersonType">
                                        <xsd:sequence>
                                        ...
                                        <xsd:sequence>
                                </xsd:complexType>
                                <xsd:element name="publisher"
                                                type="xsd:anyType"/>
                        </xsd:sequence>
                </xsd:complexType>
</xsd:schema>
```

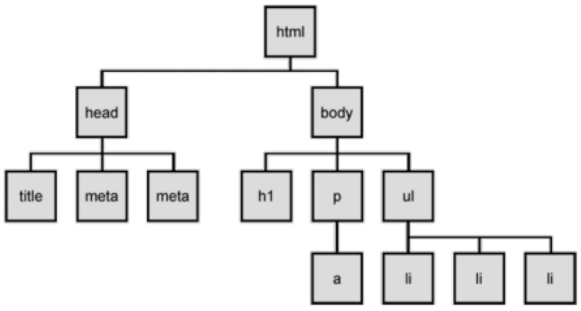
9.2 HTML5 (+DOM) + CSS3 + JS

- HTML (Hyper Text Markup Language) è un linguaggio di markup per dare struttura ai contenuti (web).
- CSS (Cascading Style Sheets) è un linguaggio per dare uno stile (di presentazione/visuale) ai contenuti (web).
- DOM (Document Object Model) è una interfaccia neutrale rispetto al linguaggio di programmazione e alla piattaforma utilizzata per consentire ai programmi l'accesso e la modifica dinamica di contenuto, struttura e stile di un documento (web). DOM è una API definita dal W3C (e implementata da ogni browser moderno) per l'accesso e la gestione dinamica di documenti XML e HTML

HTML può essere costituito da tanti tag

```
<!DOCTYPE html>
<html>
<head>
  <title>Example ...</title>
  <meta charset="UTF-8">
  <meta name="author" content="John Doe">
</head>
<body>
  <h1>Heading 1</h1>
  <p>This is a paragraph of text with a
```

```
    <a href="/path/page.html">link in it</a>.
  </p>
  <ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
  </ul>
</body>
</html>
```



Ogni **nodo** può essere caratterizzato da attributi che ne facilitano l'identificazione, la ricerca, la selezione

- Un **identificatore univoco** (attenzione: il DOM non garantisce l'unicità)
- Una **"classe"** che indica l'appartenenza ad un insieme che ci è utile definire

Il browser stesso fornisce funzionalità di ricerca:

- getElementById(IdName)
- getElementsByClassName(ClassName)
- etc.

```
<body>
        <h1>A One Page FAQ</h1>
        <div id="introText">
                <p>bla bla bla</p>
        </div>
</body>

<body>
        <h1>I've heard that JavaScript is the long-lost fountain of youth. Is this true?</h1>
        <div class="answer">
                <p>bla bla bla</p>
        </div>
</body>
```

9.2.1 CSS

Selectors

I selettori in CSS sono utilizzati per selezionare e stilizzare elementi specifici in un documento HTML. Un selettore definisce quali elementi dell'HTML verranno influenzati dalle regole di stile specificate nel blocco di dichiarazione.

I principali selettori sono

- tag name: il semplice nome del tag
  `p { ... }` //affects to all `<p>` tags

- dot (.): applicabile a un tag, indica una classe
  `p.highlight { ... }` //affects all `<p>` tags with class="highlight"
- sharp character (#): applicabile a un tag, indica un identificativo
  `p#intro { ... }` //affects to the `<p>` tag with the id="intro"
- two dots (:): stati comportamentali (ad esempio evento mouseover)
  `p:hover { ... }` //affects to `<p>` tags with the mouse over
- brackets ([attr='value']): tag con un valore specifico per un attributo 'value'
  `input[type="text"] {...}` // affects to the input tags of the type text

Media query

Le media query possono essere viste come particolari selettori capaci di valutare le capacità del device di accesso alla pagina.
Si possono controllare ad esempio:

- Larghezza e altezza (width, height) del device o della finestra
- Orientamento dello schemo (landscape/portrait)
- Risoluzione

Se la pagina è più larga di 480 pixel (e si sta visualizzando sullo schermo), applica determinati stili agli elementi con id “leftsidebar” (un menu) e “main” (la colonna centrale).

Cascading

Esistono potenzialmente diversi stylesheet:

- L’autore della pagina in genere ne specifica uno (il modo più comunemente inteso) o più d’uno
- Il browser ne ha uno, o un vero e proprio CSS o simulato nel loro codice
- Il lettore, l’utente del browser, ne può definire uno proprio per customizzare la propria esperienza

Dei conflitti sono quindi possibili (inevitabili), ed è necessario definire un algoritmo per decidere quale stile vada applicato a un elemento.

L’impostazione di HTML e CSS separa nettamente il contenuto dalla modalità di visualizzazione

- “Separation of concerns” è un principio di design altamente desiderabile in contesto informatico
- Purtroppo lo stile invece può avere un forte coupling con il contenuto, come risulta dagli esempi precedenti

9.2.2 JSON

> **Info**
>
> JSON (JavaScript Object Notation) è un formato leggero per lo scambio di dati facile da leggere e scrivere per gli esseri umani, da analizzare e generare per le macchine.
> È un formato di testo completamente indipendente dal linguaggio e utilizza convenzioni familiari ai programmatori della famiglia dei linguaggi C, tra cui C, C++, C#, Java, JavaScript, Perl, Python e molti altri.

JSON si basa su due strutture dati universali:

- Una **collezione di coppie nome/valore**. In vari linguaggi, questo è realizzato come oggetto, record, struct, dizionario, tabella hash, elenco di chiavi o array associativo.
- Un **elenco ordinato di valori**. Nella maggior parte dei linguaggi, viene realizzato come array, vettore, elenco o sequenza.

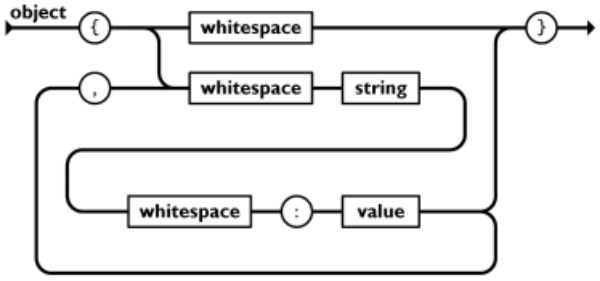
Praticamente tutti i moderni linguaggi di programmazione li supportano in una forma o nell'altra.

Tipi base:

- Numeri
- String
- Booleani
- Array
- Oggetti
- Null

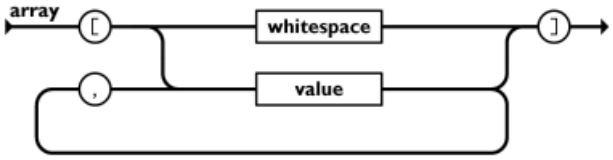
JSON objects

- Insieme non ordinato di coppie nome/valore
- Inizia con { e termina con }
- Ogni nome è seguito da :
- Le coppie nome/valore sono separate da ,



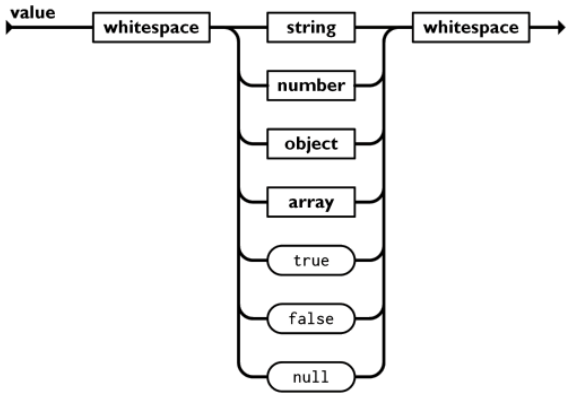
JSON array

- Insieme ordinato di valori
- Inizia con [e termina con]
- I valori sono separati da una virgola



JSON value

- Un valore può essere una stringa tra doppi apici, o un numero, o vero o falso o nullo, oppure un oggetto o un array.
- Le strutture di oggetti e array possono essere annidate.



JSON string

- Sequenza di zero o più caratteri Unicode, avvolti in "doppi virgolette, utilizzando \
- Un carattere è rappresentato come una singola stringa di caratteri
- Una stringa è molto simile a una stringa in C o Java.