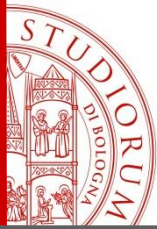


HTML5

prima parte

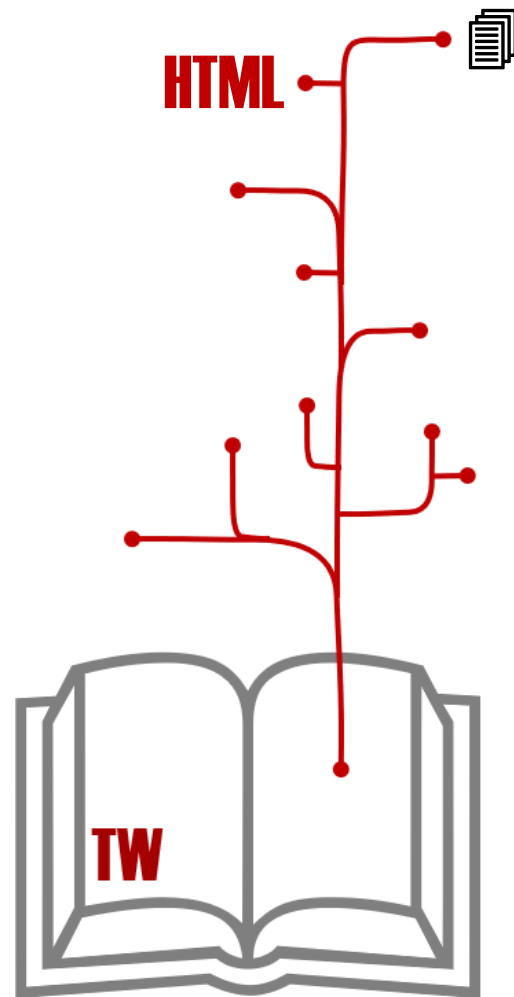


lezione due



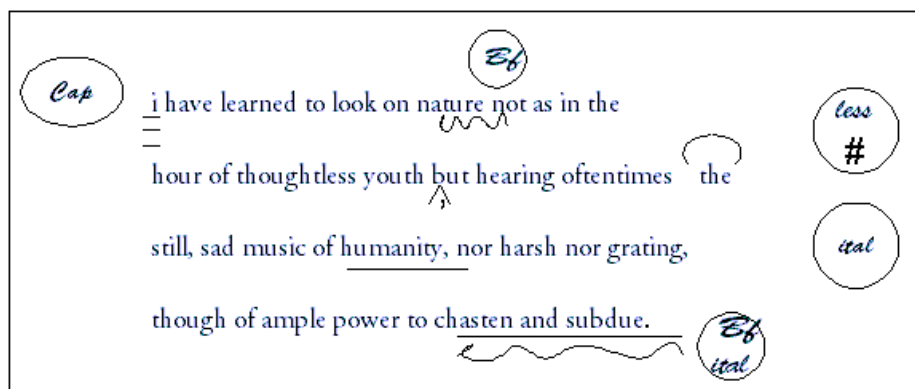
Argomenti

- HTML:
 - Definizione di Markup e XML
 - HTML e standardizzazione
 - Introduzione a HTML5
 - Categorie e content model HTML5
 - Metadata
 - Codifica dei caratteri



Markup

- Un **linguaggio di markup** è un linguaggio (con una specifica **sintassi**) che consente di **annotare** un documento fornendone una interpretazione delle sue parti
- Il termine markup (o **marcatura**) deriva dal contesto della tipografia, dove si usava marcare con annotazioni le parti del testo che andavano evidenziate o corrette





Tipi di markup

- Procedurale/descrittivo:
 - i linguaggi di markup di tipo **procedurale** indicano le **procedure di trattamento** del testo, il markup specifica le istruzioni che devono essere eseguite per visualizzare la porzione di testo referenziata (es. **TEX**)
 - i linguaggi di markup di tipo **descrittivo** identificano strutturalmente il tipo di ogni elemento del contenuto. Invece di specificare effetti grafici come l'allineamento o l'interlinea, ne **individuo il ruolo all'interno del documento**, per esempio specificando che un elemento è un titolo, un paragrafo, o una citazione, ecc. (es. **HTML, XML, SGML, ...**)



Metamarkup e XML

- Il **metamarkup** consiste nel fornire regole di interpretazione del markup e permette di definire nuovi linguaggi di markup

Un metalinguaggio di markup è un linguaggio che serve a definire altri linguaggi di markup. In altre parole, permette di creare le regole e la sintassi per costruire linguaggi specializzati, personalizzabili in base al tipo di documenti o dati da rappresentare.

- Un **metalinguaggio di markup**:
 - fornisce una sintassi: definire le regole da applicare nella marcatura di un determinato tipo di documenti.
 - consente di **definire altri linguaggi di markup** descrivendone la grammatica.
- **XML** (Extensible Markup Language) è un linguaggio di markup, progettato per lo **scambio e la interusabilità di documenti strutturati su Internet**

XML e Document publishing



poiché XML separa la struttura del documento dal suo formato di visualizzazione, un documento in XML può essere utilizzato per diversi scopi senza modificarne il contenuto.

- Definizione di **documenti strutturati**:

- XML è ideale come linguaggio per **definire documenti strutturati o semi strutturati**, e per esprimerli in maniera indipendente dalla loro destinazione finale
- L'XML permette di definire e utilizzare elementi e attributi personalizzati definendo in questo modo **nuovi linguaggi di markup specifici**
- Lo stesso documento XML può essere preso e **trasformato** per l'editing, la stampa, il Web, lo smartphone,

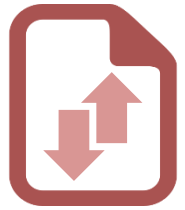
XML consente di definire elementi e attributi che riflettono esattamente le esigenze del documento in questione. Ad esempio, per una fattura, possiamo avere elementi come <cliente>, <data>, <totale> con attributi che danno più dettagli (come valuta="euro" per il totale).

XML può essere trasformato e formattato per l'output desiderato. Ad esempio:

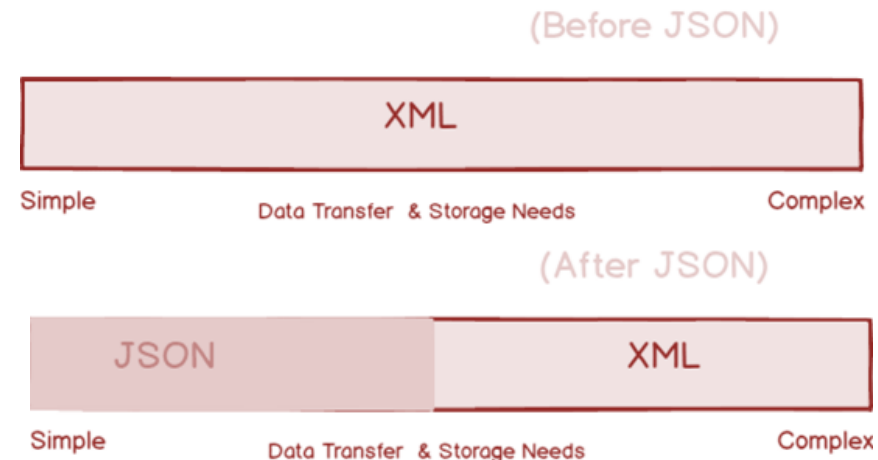
- Per la stampa, il documento XML può essere trasformato in un formato di stampa (come PDF) con una struttura e uno stile specifici.
- Per il web, lo stesso documento XML può essere trasformato in HTML, rendendolo leggibile su un browser.



XML e data management



- Nella visione W3C XML era anche un mezzo generale di descrizione del formato dei dati per:
 - **Comunicazione**, realizzazione di **protocolli** che scambiano dati in formato XML
 - **Memorizzazione**, realizzazione di DB (integrati o no con il modello relazionale) basati su formati XML
- La tendenza attuale delle tecnologie web (per es. JSON, MongoDB) va in altre direzioni...





XML

- XML **fornisce una sintassi** con cui è possibile specificare gli elementi e gli attributi che possono essere utilizzati all'interno di particolari classi di documenti
- Utilizzando XML è possibile creare un modello, chiamato **Document Type Definition (DTD)**, che descrive la struttura e il contenuto di una classe di documenti
- Lo stesso **XML ha un proprio DTD** (REC-xml-19980210) in cui vengono elencate le regole della specifica del linguaggio. Anche HTML prima della versione 5 era definito tramite DTD ...



Well-Formed vs Valid

- In XML si possono generare documenti:
 - **Well-Formed** (**Ben formati**) conformi alle generiche specifiche XML. Occorre solo definire un file XML che le rispetti.
 - **Valid** (**Validi**), conformi ad una specifica DTD. Occorre definire un file XML e la sua DTD. La DTD può essere sia **interna** (nel file xml) che **esterna** (in un altro file). Il documento appartiene ad una classe di documenti, definiti dalla specifica DTD. Un documento valido deve essere sempre e comunque ben formato.



Esempio

```
<!-- Intestazione -->
```

```
<?xml version="1.0" ?>
```

```
<!-- DTD -->
```

```
<!DOCTYPE tesi SYSTEM "tesi.dtd">
```

VALIDO

elemento

attributo

```
<tesi sessione='II'>
```

```
<titolotesi> Linguaggi di Markup</titolotesi>
```

```
<autore> Pinco Pallino </autore>
```

```
<relatore> Paola Salomoni</relatore>
```

```
<capitolo>
```

```
<numero>1</numero>
```

```
<titolo>introduzione</titolo>
```

```
<paragrafo>
```

```
. . . . .
```

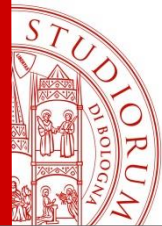
```
</paragrafo>
```

```
</capitolo>
```

```
</tesi>
```

BEN FORMATO

perché chiusura dei tag



L'elemento <tesi> è un contenuto strutturato perché contiene una gerarchia ordinata di altri elementi interni

Esempio

Definisce un elemento chiamato <tesi> che contiene altri elementi interni (contenuto strutturato). Al suo interno deve avere obbligatoriamente <titolotesi>, <autore>, <relatore>, <capitolo>.

Esempio di **tesi.dtd**

```
<!--Dichiarazione della DTD -->
```

```
<!ELEMENT tesi (titolotesi, autore, relatore, correlatore?, capitolo+)>
```

```
<!ATTLIST tesi sessione ('I' | 'II' | 'III') #REQUIRED>
```

L'elemento <tesi> ha un attributo chiamato sessione ed è obbligatorio

```
<!ELEMENT autore (#PCDATA)>
```

```
<!ELEMENT relatore (#PCDATA)>
```

```
<!ELEMENT correlatore (#PCDATA)>
```

```
<!ELEMENT capitolo (numero, titolo, paragrafo+)>
```

```
<!ELEMENT paragrafo (#PCDATA | sottoparagrafo+)>
```

```
<!ELEMENT sottoparagrafo (numero, titolo, paragrafo+)>
```

Contenuto Strutturato

Contenuto
strutturato

Ordine

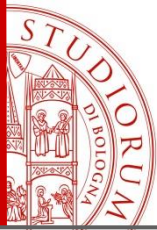
? (0 o 1 volta)
+ (1 o più volte)
* (0 o più volte)

Stringa di caratteri
alfanumerici

Contenuto testua-
le non strutturato

Attributo
obbligatorio

Contenuto
misto

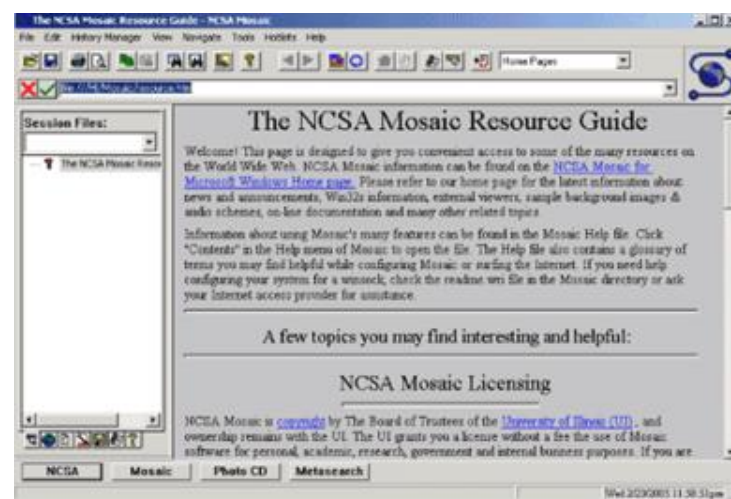


Dal primo HTML a HTML5

- Quando nasce il WWW, Tim Berners Lee e gli altri ricercatori del CERN che ne definiscono il funzionamento mandano all'IETF le specifiche del **protocollo di livello applicazione** (HTTP, 1990) e del sistema di identificazione delle risorse (URI/URL, 1994).
- La prima versione di HTML viene anche essa proposta all'IETF che però produce il primo standard (**rfc1866, Hypertext Markup Language - 2.0**) solo nel 1995.
- Il processo di standardizzazione usato per Internet si dimostra ~~però~~ inadeguato a seguire l'evoluzione di HTML



- Dopo il primo prototipo presentato dal CERN nel **1991**, nell'ottobre del **1992** il National Centre for Supercomputing Applications (NCSA) decise di realizzare una versione propria di WWW, **Mosaic**.
- Mosaic fu il primo browser a ottenere successo su larga scala e guidò lo sviluppo del web come killer application.
- Marc Andreessen, realizzatore del prototipo di Mosaic su Windows, fondò la Mosaic Corporation, poi rinominata **Netscape**, insieme a Jim Clark di Silicon Graphics.

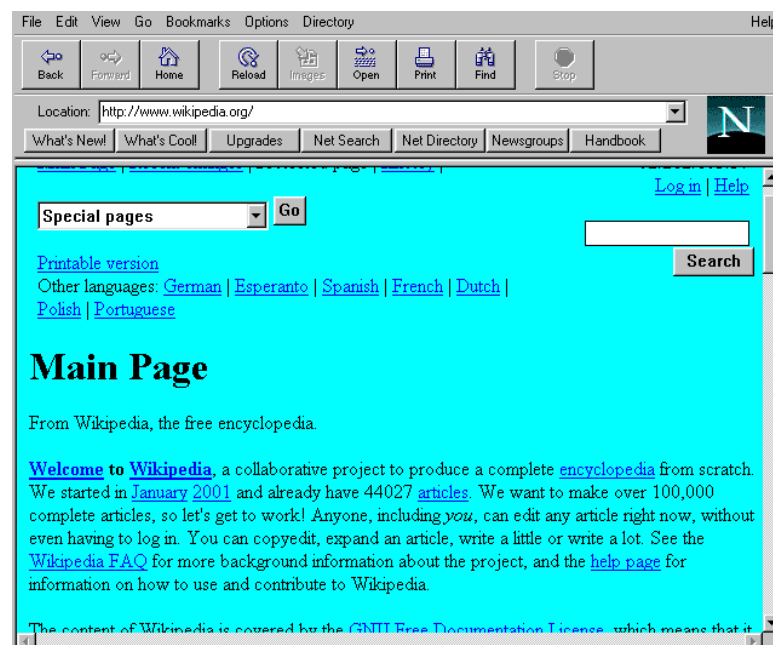


Netscape



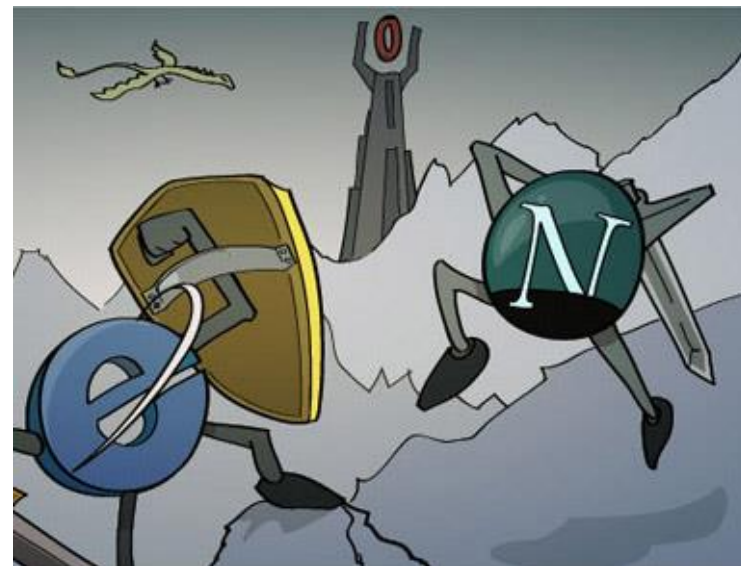
NETSCAPE®

- Nel 1994 esce **Netscape Navigator** che ha subito un successo assoluto e diventa rapidamente il browser più utilizzato:
 - La Netscape ha il passaggio più rapido tra la fondazione e la quotazione in borsa della storia, ed una delle quotazioni iniziali di maggior successo
 - Netscape diventa il browser più diffuso e lavora per mantenere competitività e controllo del mercato



La prima guerra dei browser

- Microsoft, dopo una falsa partenza con Microsoft Network, abbraccia definitivamente e con energia la tecnologia Internet, e realizza un browser (**Internet Explorer, 1995**) ed un server (Microsoft Information Server)
- Sia Netscape che Microsoft introducono piccoli miglioramenti su HTML per migliorare l'esperienza dell'utente e diffondere maggiormente il proprio browser (Introducono Tag Proprietari)
- **Opera**, uscito anche esso nel 1994 resta sullo sfondo ...



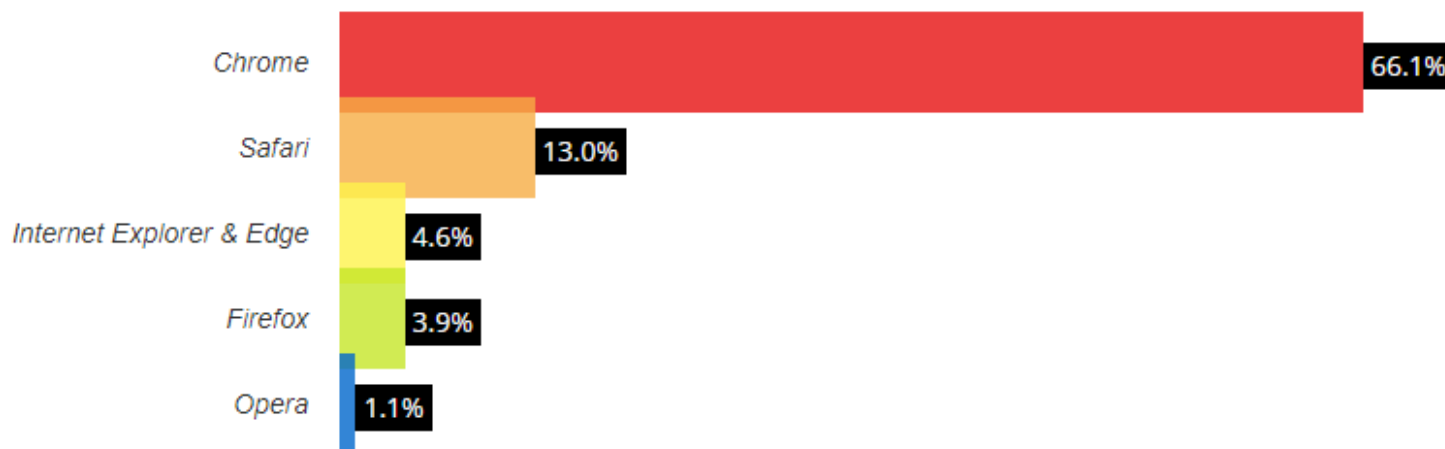


Fine della prima guerra

- Alla fine dello scontro prevalse la **Microsoft** perché scelse di distribuire il proprio browser, **Internet Explorer 3**, includendolo in Windows 95: cambiarlo diventò così un compito degli utenti, che nella maggior parte dei casi non lo fecero.
- Microsoft fu condannata nel 1997 per posizione dominante, continuando però a imporre precise specifiche che limitavano l'installazione di software di terze parti con il SO.
- Nel marzo **1998 Netscape ammette la disfatta**, rilasciando il codice sorgente della versione 5 di Navigator in open source e cercando di creare una comunità online.
- Nasce mozilla.org (prima versione di **Mozilla Firefox 2002**).

Seconda guerra dei browser

- A partire dal 2004 iniziarono ad affermarsi nuovi browser gratuiti (o addirittura open source) dotati di caratteristiche innovative e di un maggiore rispetto degli standard.
- Inizia la **seconda guerra dei browser**, che vede Explorer perdere completamente la propria egemonia a favore di **Google Chrome**, uscito nel **2008** è il browser più usato al mondo da aprile 2016.

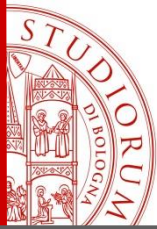


<https://www.w3counter.com/globalstats.php>

La fine di un'era

- **Microsoft ha terminato il supporto a Internet Explorer 11** nelle app e nei servizi Microsoft 365 il **17 agosto 2021**, con ritiro definitivo a **giugno 2022**
- L'ultima versione di IE non è quindi più supportata dai servizi online di Microsoft (Office 365, OneDrive, Outlook)
- Il 30 novembre 2020 è **terminato il supporto di IE 11** sulla web app di Microsoft Teams





Uso dei browser oggi

- Fonte W3Schools' Famous Month-by-Month Browser Statistics (Since 2002),
<http://www.w3schools.com/browsers/default.asp>

2024	Chrome	Edge	Firefox	Safari	Opera
March	77.6 %	10.7 %	4.6 %	3.7 %	2.2 %
February	77.5 %	10.5 %	4.6 %	3.6 %	2.0 %
January	78.1 %	10.4 %	4.7 %	3.8 %	2.1 %
2023	Chrome	Edge	Firefox	Safari	Opera
December	78.2 %	10.0 %	4.6 %	3.7 %	2.1 %
November	77.4 %	10.6 %	4.9 %	3.9 %	2.4 %
October	78.0 %	10.3 %	4.8 %	3.9 %	2.3 %
September	78.8 %	10.3 %	4.6 %	3.4 %	2.2 %
August	80.1 %	9.8 %	4.6 %	3.0 %	1.8 %
July	79.9 %	10.0 %	4.8 %	3.0 %	1.8 %

- L'effetto della battaglia sull'HTML nella prima guerra dei browser è una perdita di adesione allo **standard**:
 - Ci sono pagine compatibili solo con l'HTML di IE e altre solo con quello di Netscape
 - Il processo di standardizzazione è di importanza centrale e serve un presidio diverso da IETF
 - Berners-Lee e Cailliau fondano il **W3C (World Wide Web Consortium)**, con fondi della ricerca e dell'università. Berners-Lee viene nominato **presidente a vita**.
 - Successivamente, il W3C ha diretto lo sviluppo dei più importanti standard del Web: URI, HTTP, CSS, XML, WAI sull'accessibilità.
 - Tra il 1995 e il 1997 furono prodotte Recommendation che standardizzavano diverse estensioni dell'HTML originario, fino ad arrivare alla specifica di **HTML 4**.



nel 1994

WHAT WG

- Nel **2004**, Firefox e Opera proposero al W3C la riapertura del Working Group su HTML per lo sviluppo di nuove versioni del linguaggio. La proposta, ignorando volutamente XHTML e la rigida sintassi di XML, venne **bocciata dal W3C**.
- Venne allora formato un gruppo separato, chiuso e finanziato dalle società di software, il **Web Hypertext Application Technology (WHAT WG)** che sviluppò proposte (Web Application 1.0) che vennero effettivamente implementate da vari browser. Queste modifiche riguardavano HTML, CSS, DOM e Javascript, e cambiavano radicalmente alcuni aspetti di stretta competenza del W3C.



Welcome to the WHATWG community

Maintaining and evolving HTML since 2004



HTML, the living standard

- Nonostante la presenza di una Recommendation (*This specification defines the 5th major revision of the core language of the World Wide Web: the Hypertext Markup Language, HTML*), la specifica di HTML è una attività che WHAT WG sta continuando definendo HTML come **standard vivente**.
- Il fatto che HTML non si stabilizzi è considerato **una feature, non un bug**.

WHATWG

HTML: The Living Standard

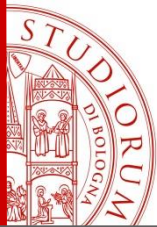
A technical specification for Web developers

Il fatto che HTML sia un "Living Standard" significa che non viene più considerato come una versione "finale" o statica, ma piuttosto come uno standard in continua evoluzione, costantemente aggiornato e migliorato. Questo è diverso dal concetto tradizionale di una "Recommendation" del W3C, che in passato indicava una versione stabile e definitiva di uno standard.



HTML, the living standard

- Il WHAT WG non si propone di stabilizzare una specifica:
 - *Because the specification is now a living document, we are today announcing two changes:*
 - *The HTML specification will henceforth just be known as "HTML".*
 - *The WHATWG HTML spec can now be considered a "living standard". It's more mature than any version of the HTML specification to date, so it made no sense for us to keep referring to it as merely a draft.*



HTML, the living standard

WHATWG HTML Living Standard: è lo standard "vivente" che evolve in tempo reale, e rappresenta lo stato più aggiornato di HTML. I browser moderni si allineano a questo standard. W3C "forks": sono documenti separati, estratti dallo standard del WHATWG, che talvolta presentano delle differenze. Questi fork possono essere visti come versioni stabili, ma spesso non riflettono tutte le ultime modifiche o aggiunte dell'HTML Living Standard.

- La relazione tra lo sviluppo continuo del WHAT WG e la standardizzazione operata dal W3C non è semplice:
 - *The W3C also publishes parts of this specification as separate documents that are forked subsets of the “HTML Living Standard”. There are numerous differences between the HTML Living Standard and the W3C forks; some minor, some major.*
- Così facendo **non c'è più una versione di riferimento** con cui confrontare le funzionalità dei browser e per gli sviluppatori verificare la conformità allo standard è molto complesso.
- Quello che si può verificare è che l'applicazione giri su tutti i browser ...

Il focus principale per gli sviluppatori è garantire che le applicazioni siano compatibili con tutti i principali browser, invece di fare riferimento a una specifica versione di HTML.

Poiché HTML è in continua evoluzione come standard vivente, non esiste una singola versione definitiva o "stabile" con cui confrontare l'implementazione di un browser. Questo rende complesso per gli sviluppatori verificare se un browser è completamente conforme a un particolare standard.







«Nostra» prospettiva

- Noi scriveremo codice HTML con una **prospettiva specifica**, con l'obiettivo di:
 - Scrivere codice **well formed** (XHTML), ovvero ordinato
 - Seguire la **semantica degli elementi** (Ogni elemento ha un ruolo)
 - Separare la **presentazione dal contenuto**
 - Scrivere codice **accessibile**
- Ci sono altre prospettive importanti:
 - Quella della **portabilità** del codice
 - Quella della **velocità di realizzazione** (spesso tradisce le altre)
 - ...
- **Occhio che questi principi DEVONO essere applicati anche durante l'esame**





Noi, XHTML



- Anche se la sintassi standard non forza la compliance di HTML5 a XML, noi cerchiamo sempre di scrivere HTML5 **well formed**, ovvero:
 - Elementi tutti in minuscolo:
 -  • Sì: `<html>`
 -  • No: `<HTML>`
 - Elementi con contenuto aperti e chiusi, ed elementi vuoti (senza contenuto) scritti correttamente:
 -  • Sì: `<html>` con `</html>` e `
`
 -  • No: `<html>` senza `</html>` e `
`



Noi, separare

- **Separiamo bene:**
 - il contenuto (che è specificato dal markup HTML5)
 - dagli aspetti presentazionali, come la resa grafica (specificato dal foglio di stile, con CSS3).
- Esempio:
 -  – Sì: Uso un elemento **** per dare caratteristiche più evidenti ad una porzione di testo che non è titolo.
 -  – No: uso un elemento **<h4>** per dare caratteristiche più evidenti a una porzione di testo che non è titolo.
- Questo principio era applicabile anche con le precedenti versioni di HTML ma la maggiore strutturazione semantica del documento lo rende più agile.

Noi, **strutturare**

- HTML5 ha nuovi elementi:
 - che consentono di **strutturare** il contenuto attribuendo alle parti di documento una **semantica**. La pagina deve quindi essere divisa in parti a secondo del ruolo che queste hanno e questi ruoli (header, footer, ecc.) devono essere specificate.
 - che definiscono bene elementi di controllo, menù e strumenti di navigazione
- Noi li useremo più possibile:
 -  – Si: Uso un elemento `<article>` per indicare il contenuto specifico di un post di un blog.
 -  – No: il contenuto specifico di un post di un blog lo inserisco in un paragrafo `<p>` generico, eventualmente distinto graficamente via CSS.



Noi, accessibile

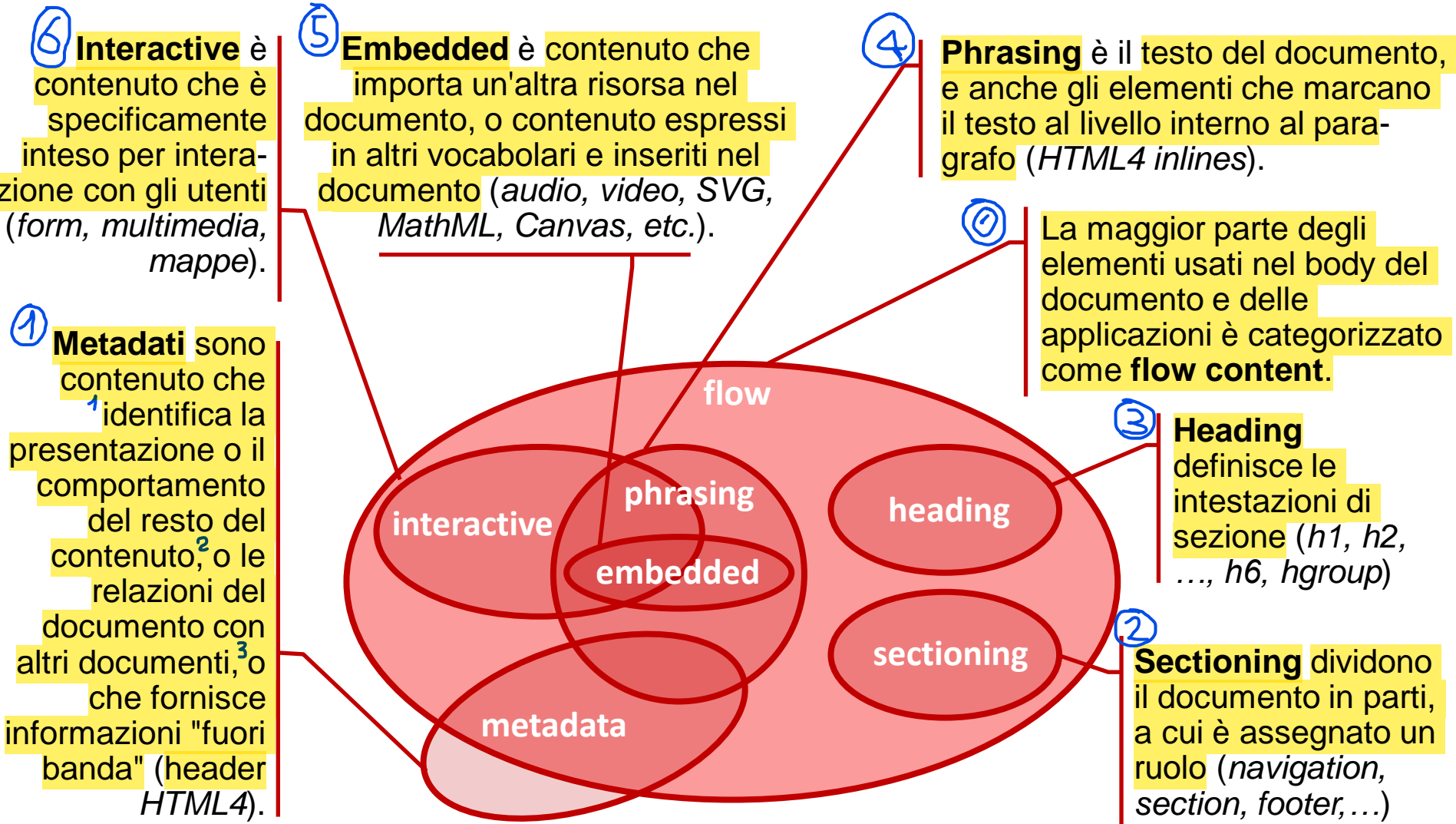
- **Accessibilità** è *"la capacità dei sistemi informatici, nelle forme e nei limiti consentiti dalle conoscenze tecnologiche, di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive o configurazioni particolari"*
- Ne parleremo meglio in future lezioni e in un seminario e specificheremo le linee guida per produrre pagine accessibili (i casi sì e no per l'accessibilità).
- **Per il momento introdurremo gli elementi e gli attributi HTML5 che consentono di realizzare pagine accessibili.**



Elementi HTML5

- Ogni **elemento** HTML5 fa parte di una o più **categorie**, definiti per gruppi con caratteristiche simili:
 - Metadata, Flow, Sectioning, Heading, Phrasing, Embedded, Interactive.
- Le categorie per la Recommendation W3C sono riportati qui (con grafica interattiva):
<http://www.w3.org/TR/html5/dom.html#kinds-of-content>
- Si noti che:
 - Altre categorie/sottocategorie possono essere usate per raggruppare elementi con scopi specifici (per esempio i controlli delle form).
 - Alcuni elementi possono avere caratteristiche uniche e non appartenere a nessuna categoria.
 - **Non usiamo la classificazione in modo rigido ma come strumento per introdurre i principali elementi**

Elementi e categorie





Prima pagina

- Un documento HTML5 è:
 - Definito dal `DOCTYPE` come html
 - Incluso tra elemento `<html>` e `</html>`
 - Strutturato in:
 - `<head></head>` intestazione del documento che riporta informazione sulla pagina o sulle relazioni con altri documenti. In questa parte è riportato il titolo che sarà mostrato nel tab della finestra del browser e il `charset` in uso.
 - `<body></body>` corpo del documento che racchiude il vero e proprio contenuto della pagina

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Nome</title>
```

```
<meta
```

```
charset="UTF-8"/>
```

```
</head>
```

```
<body>
```

```
Content of the  
document ...
```

```
</body>
```

```
</html>
```

- Gli elementi di **metadata** consentono di descrivere il documento specificandone caratteristiche, comportamento, presentazione, relazioni con altri documenti.
- I metadati sono inclusi nell'head del documento:
 - `<title>`,
 - `<base>`,
 - `<link>`,
 - `<meta>`,
 - `<style>`.

<title>

- L'elemento `<title>` rappresenta il **titolo** o il nome del documento.
 - Deve essere scritto tenendo conto che potrebbe essere usato fuori dal contesto (come per esempio accade nei bookmark) ovvero senza avere a corredo il contenuto del documento (cosa che lo differenzia da `h1`).
 - Ogni documento deve avere al massimo un elemento `<title>`.
- Esempio:

```
<head>  
  <title>Materiale dell'insegnamento di  
    Tecnologie Web - CdS ISI Cesena</title>  
</head>
```



<base>

- L'elemento **<base>** deve essere unico all'interno del documento, se ci sono più elementi **<base>**, viene preso in considerazione solo il primo.
- Lo scopo principale dell'elemento **<base>** è quello di indicare il **path base** del documento che servirà per risolvere gli URL relativi, sia in termini di **href** che di **target** (**_blank** apre una nuova finestra)
- Esempio:

```
<head>
```

```
<base href="http://www.w3schools.com/images/"  
target="_blank"/>
```

```
</head>
```

```
<body>
```

```
<base href="https://www.esempio.com/assets/" target="_blank">
```

```
<a href="documentazione.html">Doc</a> (risolto come  
https://www.esempio.com/assets/documentazione.html)
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_base_test



<link>

- L'elemento `<link>` viene usato per creare **relazioni** tra il documento e altri documenti o risorse.
- Ha molti utilizzi ma quello principale è creare la relazione con il CSS usato dal documento.
- Esempio:

```
<head>  
  <link rel="stylesheet" type="text/css"  
    href="theme.css" />  
</head>
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_link_tag

<style>

NON USARLO MAI PERCHÉ I FILE CSS
VENGONO USATI TRAMITE LINK

- L'elemento style permette di includere stili all'interno del documento.
- Il rendering del documento sarà il risultato dei <link> a fogli di stile, degli elementi <style> e degli eventuali (meglio di no!) stili inline utilizzati a cui si aggiungono gli stili dell'utente.
- Per esempio:

```
<style>
  h1 {color:red;}
  p {color:blue;}
</style>
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_style

<meta>

- I **<meta>** vengono usati per aggiungere altri **metadati** al documento. Sono spesso usati dai motori di ricerca.
- Il tipo di metadati è specificato dall'attributo **name**. In particolare il meta **description** è usato per lo snippet da Google.
- Esempio:



```
<head>
  <meta charset="UTF-8" />
  <meta name="description"
    content="Free Web tutorials" />
  <meta name="keywords"
    content="HTML, CSS, XML, JavaScript" />
  <meta name="author" content="Hege Refsnes" />
</head>
```

Aggiunge la
descrizione
con contenuto
"Free Web
Tutorials"

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_meta

DOMANDA 1:

Quale di questi elementi non indica un metadato:

☐ `<link>`

☐ `<title>`

☐ `<base>`

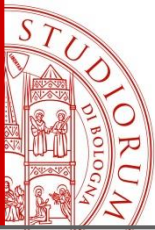
☒ `<head>`



Il codice ASCII

- ASCII (American Standard Code for Information Interchange) è una codifica:
 - Definita nel 1961
 - pubblicato dall'American National Standards Institute (ANSI) nel 1968,
 - Diventata standard ISO (ISO 646) nel 1972.
- E' una codifica basata su 7 bit: ^{e 1 di controllo} si usa un byte di memoria ma gli ottetti da 128 a 255 non sono utilizzati.

000	nul	001	soh	002	stx	003	etx	004	eot	005	enq	006	ack	007	bel
008	bs	009	ht	010	nl	011	vt	012	np	013	cr	014	so	015	si
016	dl	017	dc1	018	dc2	019	dc3	020	dc4	021	nak	022	syn	023	etb
024	can	025	em	026	sub	027	esc	028	fs	029	gs	030	rs	031	us
032	sp	033	!	034	"	035	#	036	\$	037	%	038	&	039	'
040	(041)	042	*	043	+	044	,	045	-	046	.	047	/
048	0	049	1	050	2	051	3	052	4	053	5	054	6	055	7
056	8	057	9	058	:	059	;	060	<	061	=	062	>	063	?
064	@	065	A	066	B	067	C	068	D	069	E	070	F	071	G
072	H	073	I	074	J	075	K	076	L	077	M	078	N	079	O
080	P	081	Q	082	R	083	S	084	T	085	U	086	V	087	W
088	X	089	Y	090	Z	091	[092	\	093]	094	^	095	_
096	`	097	a	098	b	099	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124		125	}	126	~	127	del



Varianti nazionali di ISO 646

- Il codice ASCII non contiene alcuni caratteri molto usati in alcune lingue europee (per esempio tutte le lettere accentate).
- In ISO 646 sono definite anche **varianti nazionali**, in cui alcune posizioni sono assegnate per uso nazionale
- Queste posizioni sono:
 - **@[\|]** sempre e
 - **#\$^`~** se necessario.

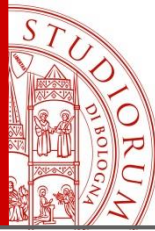
dec	hex	glifo	variante
35	23	#	£ Ù
36	24	\$	¤
64	40	@	É § Ä à ³
91	5B	[Ä Æ ° â î ÿ é
92	5C	\	Ö Ø ç Ñ ½ ¥
93	5D]	Å Ü § ê é ç
94	5E	^	Ü î è
96	60	`	é ä µ ô ù
123	7B	{	ä æ é à ° °
124	7C		ö ø ù ò ñ f
125	7D	}	å ü è ç ¼
126	7E	~	ü ¯ ß ° û ì ´ _

ISO 8859/1 (ISO Latin 1)

fa parte di

- **ISO Latin 1** è uno standard che ~~comprende~~ ISO 8859, e come tutte le specifiche ISO 8859 **utilizza 8 bit**
- I primi 128 caratteri sono quelli di ASCII, gli altri 128 sono usati per introdurre i caratteri latini specifici.
- **Copre:**
 - la maggior parte delle lingue europee occidentali: danese, faroese, finlandese, francese, gaelico scozzese, inglese, irlandese, **italiano**, norvegese, olandese, portoghese, romancio, spagnolo, svedese e tedesco.
- **Copre anche:**
 - albanese, indonesiano, afrikaans e swahili.

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0		ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
1	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
2		ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8	€		,	f	„	...	†	‡	^	%	Š	‹	Œ		Ž	
9		‘	’	“	”	•	—	~	™	š	›	œ		ž	ÿ	
A		ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı	ı
B	°	±	²	³	´	µ	¶	·	,	ı	°	»	¼	½	¾	ı
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ



Le "parti" sono codifiche individuali (sottoinsiemi) di caratteri progettate per rappresentare gruppi di lingue con esigenze di caratteri simili. Ad esempio, una parte può essere progettata per lingue dell'Europa occidentale, un'altra per lingue dell'Europa orientale, e così via.

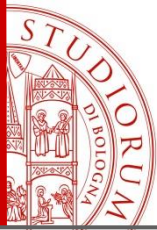
ISO 8859

ISO 8859 non è un singolo set di caratteri, ma una famiglia di codifiche di caratteri (chiamati anche "parti" o "subset"). Ogni parte copre specifiche lingue o gruppi di lingue, fornendo i caratteri necessari per scrivere in quelle lingue. Ciascuna parte è numerata e si focalizza su un insieme diverso di lingue.

- Lo standard **ISO 8859** è complessivamente composto da 16 parti, ciascuna delle quali è progettata per rappresentare lingue simili, in modo che i comuni caratteri utilizzati siano inseriti nella stessa raccolta.
- Quando un simbolo è ripetuto in più parti, generalmente mantiene la stessa posizione, in modo da limitare i problemi di conversione.
- Oltre all'ISO Latin 1, è molto usato anche l'**ISO Latin 15** che lo ha sostituito. Nella revisione:
 - Sono stati eliminati alcuni simboli scarsamente utilizzati.
 - Questi simboli sono stati sostituiti con il simbolo dell'euro € e con le lettere Š, š, Ž, ž, Œ, œ, e Ÿ, che completano la copertura di francese, finlandese ed estone.

Una revisione della 8859/1 che elimina alcuni simboli scarsamente utilizzati

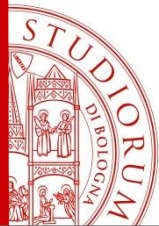
ISO Latin 1



Unicode e ISO/IEC 10646

- ISO 8859 non risolve tutti i problemi legati alle lingue con alfabeti non latini (arabo, cinese, giapponese, thailandese, ecc).
- Per affrontare in modo definitivo le questioni di internazionalizzazione si mettono al lavoro due gruppi (uno di origine commerciale, uno di origine istituzionale), che producono due standard **Unicode** e **ISO/IEC 10646**.
- I due standard sono mantenuti sincronizzati dal 1991 ma in teoria questo sodalizio potrebbe rompersi e i due standard potrebbero procedere autonomamente.





ISO/IEC 10646

Definisce un modello universale di codifica dei caratteri noto come Universal Character Set (UCS), che mira a coprire tutti i caratteri usati nelle lingue del mondo, oltre a simboli, segni e altri caratteri speciali.

- ISO/IEC 10646 definisce:

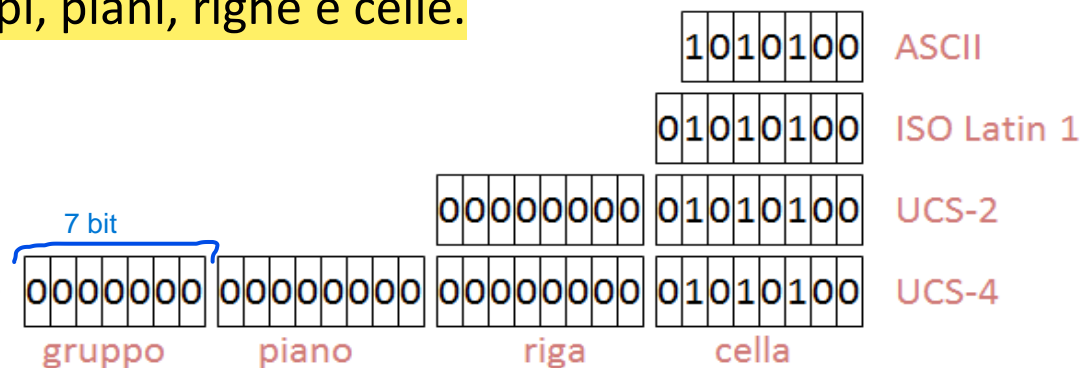
- 128 gruppi (groups) di
- 256 piani (planes) di
- 256 righe (rows) odif
- 256 celle (cells)

che potenzialmente identificano 2.147.483.648 caratteri (in realtà può codificare 679,477,248 caratteri)

- ISO 10646 è composto di due schemi di codifica.

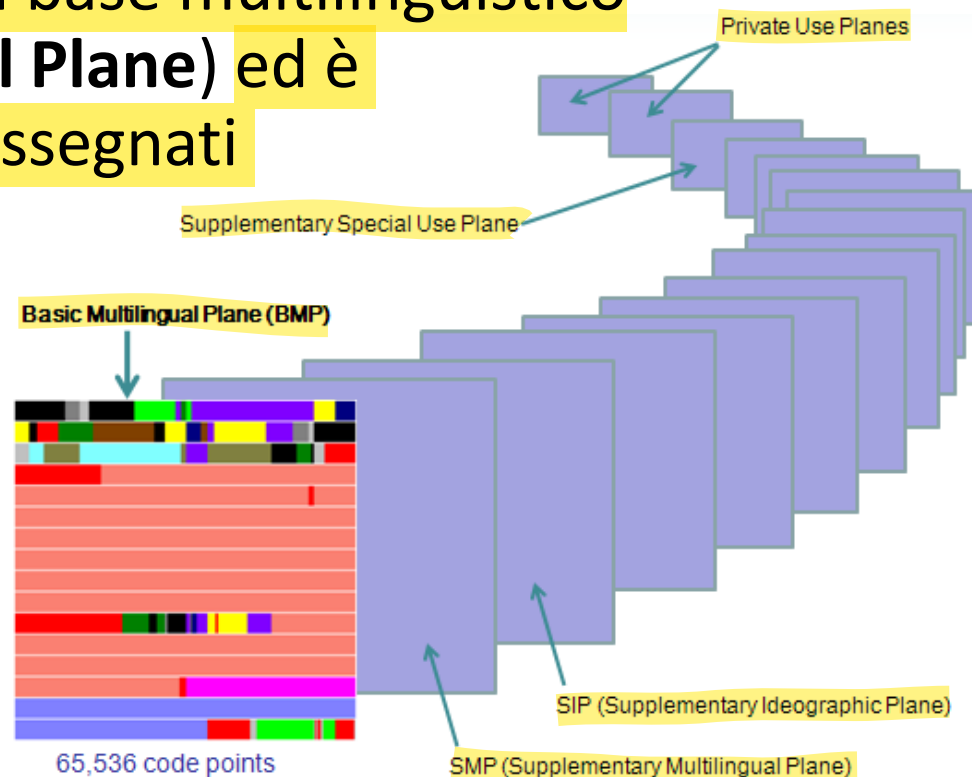
- UCS-2 è uno schema a due byte, che è un'estensione di ISO Latin 1.
- UCS-4 è uno schema a 31 bit in 4 byte, estensione di UCS-2. E' diviso in gruppi, piani, righe e celle.

- UCS-4, 31 bit per ogni carattere, che consente la codifica semplice di tutti i caratteri;
- UCS-2, due byte per ogni carattere, che consente la codifica del primo piano, 0x20, il piano multilingue di base, contenente i primi 36.864 punti di codifica, in modo semplice, e di altri piani e gruppi passando ad essi con le sequenze di escape ISO/IEC 2022;



ISO/IEC 10646: piani

- Dei 17 piani, ciascuno in grado di codificare 65.536 caratteri, sono assegnati solo i primi 3 e gli ultimi tre.
- Il piano 0 è detto piano di base multilinguistico (**BMP** - **Basic Multilingual Plane**) ed è il piano in cui sono stati assegnati la maggior parte dei caratteri.
- BMP contiene caratteri per quasi tutti i moderni linguaggi e un grande numero di caratteri speciali





UCS e UTF

- Unicode e ISO/IEC 10646 utilizzano 4 byte per la codifica di un solo carattere:
 - Risolvono i problemi di codifica delle lingue non europee MA
 - Consumano molta memoria.
- In realtà la maggior parte degli alfabeti sta nel BMP, e la maggior parte dei documenti sono scritti in ASCII.
- Quindi, al posto di UCS si usa **UTF (UCS Transformation Format)**, che consente di usare tutti i caratteri definiti in UCS ma utilizzando una codifica a lunghezza variabile.



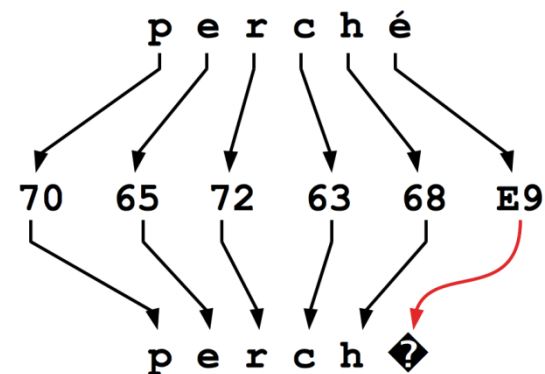
UTF-16 e UTF-8

- UTF-16: considera tutti i caratteri di UCS-2 (o in 16 bit).
- UTF-8 considera di accedere a tutti i caratteri di UCS-4, ma utilizza un numero compreso tra 1 e 4 byte per farlo.
 - I codici compresi tra 0 - 127 (ASCII a 7 bit), e richiedono un byte (sempre 0 al primo bit).
 - I codici derivati dall'alfabeto latino e tutti gli script non-ideografici richiedono 2 byte.
 - I codici ideografici (orientali) richiedono 3 byte.
 - I codici dei piani alti richiedono 4 byte.

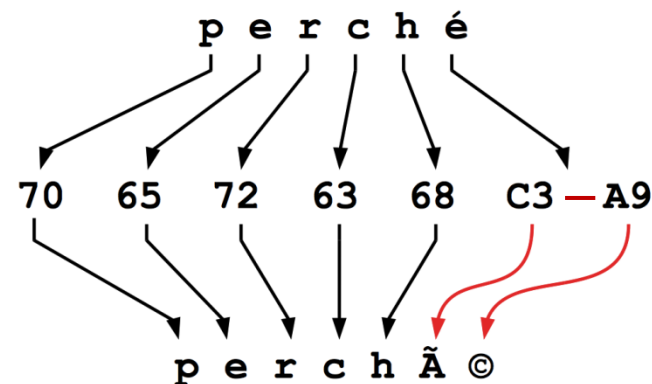
UTF-8 e Latin 1

- Le codifiche UTF-8 e Latin 1 sono compatibili ma non identiche.

- Ci possono essere problemi:
 - Aprendo un Latin 1 come UTF8:



- Aprendo un UTF8 come Latin 1:



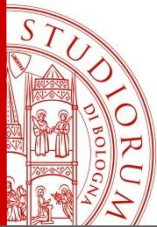
seconda domanda

**BONUS**

DOMANDA 2:

Il carattere «à»:

- ☐ Usa 8 bit per la codifica ASCII
- ☐ Usa 8 bit per la codifica Unicode
- ☒ Usa 8 bit per la codifica ISO Latin
- ☐ Usa 8 bit per la codifica UTF-8



Riferimenti

- Vedi piattaforma
- Standard W3C:
<https://html.spec.whatwg.org/multipage/>
- Living Standard:
<http://www.w3.org/TR/html5/dom.html#kinds-of-content>

