



HTML5

INTRODUCTION & SEMANTICS



POLITECNICO
DI TORINO

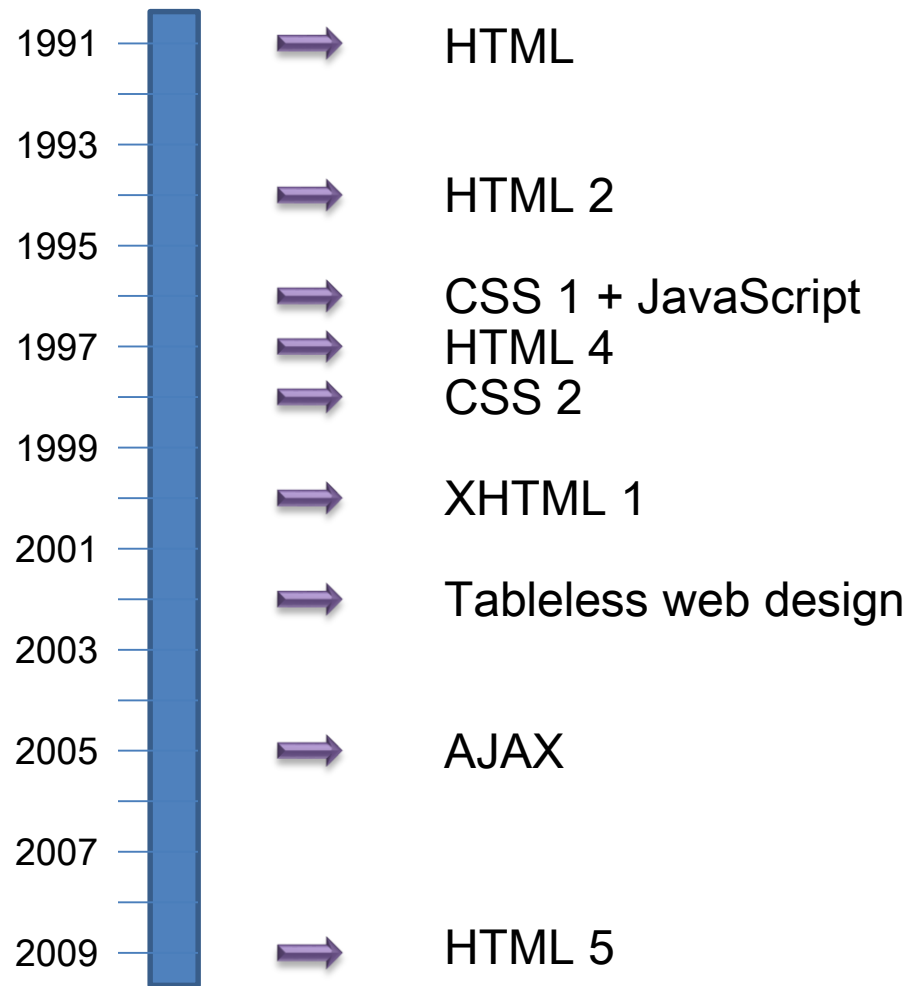


HTML5



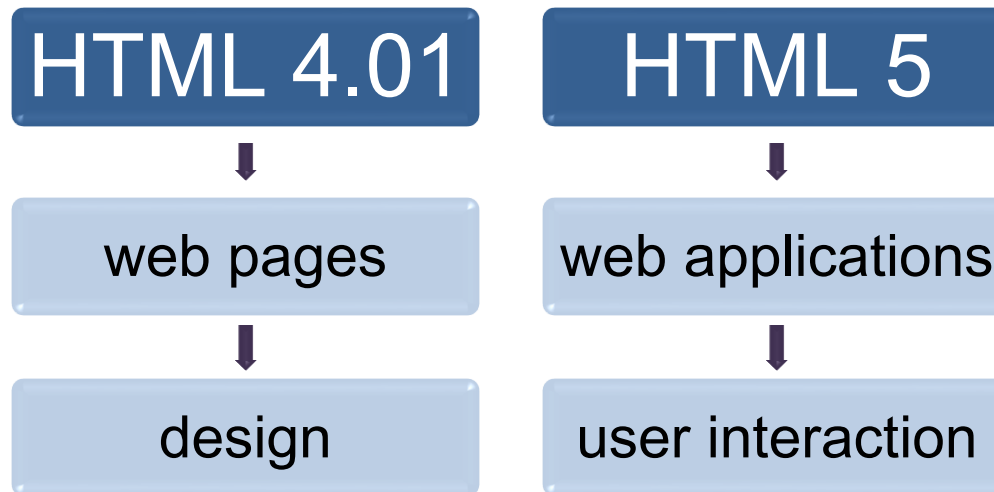
- “A vocabulary and associated APIs for HTML and XHTML”
- HTML5 is the last major revision of the Hypertext Markup Language (HTML) standard
 - W3C Recommendation 28 October 2014
 - Follows its predecessors HTML 4.01 and XHTML 1.1
 - Work on the specifications started in June 2004
- HTML5: “A vocabulary and associated APIs for HTML and XHTML”
- Currently being carried out in a joint effort between the W3C HTML WG and the WHATWG (Hypertext Application Technology Working Group)

Rough timeline of web technologies



HTML5

- HTML5 \approx HTML + CSS + JS API + DOM

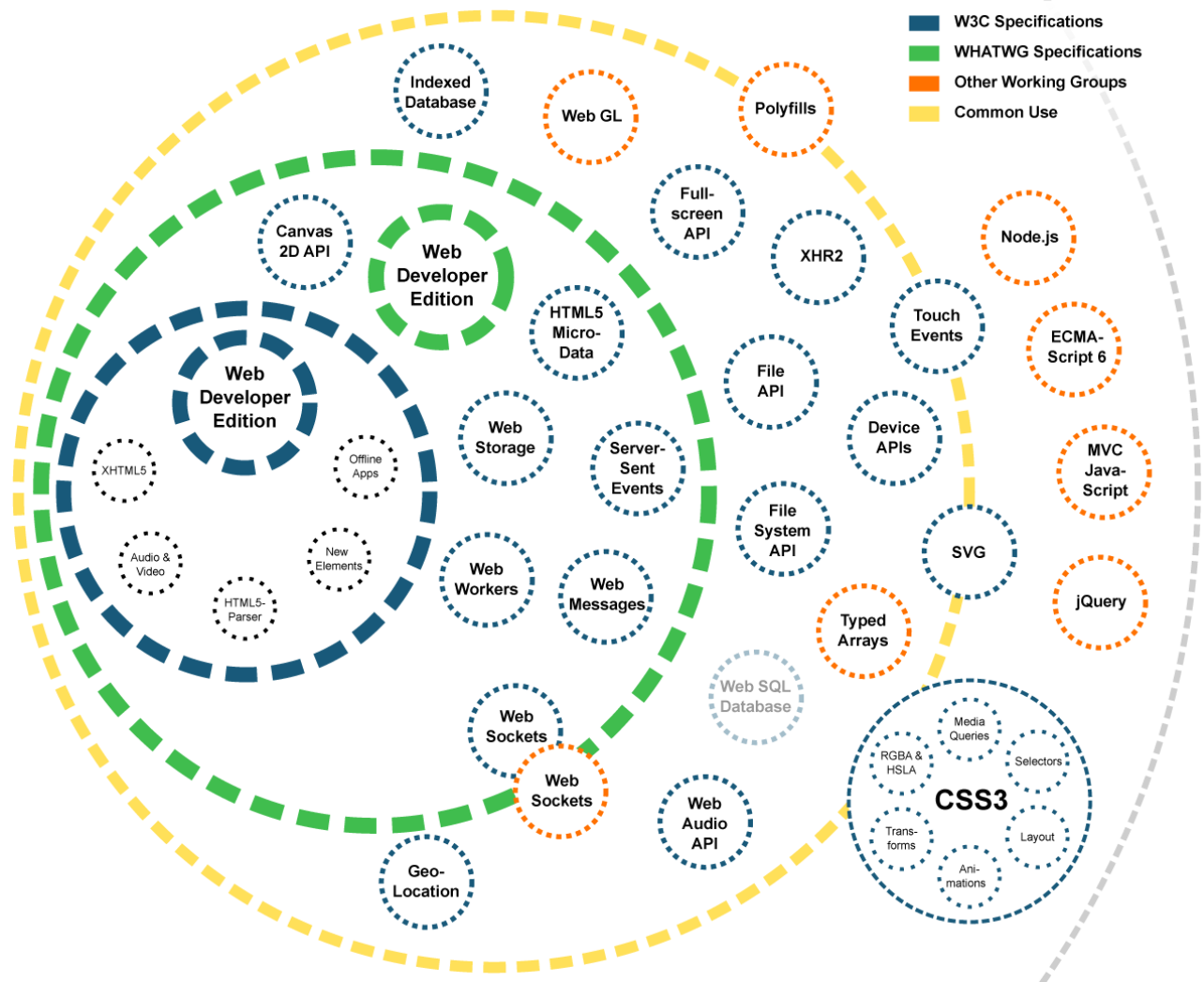


Rationale

- “This specification defines the 5th major revision of the core language of the World Wide Web: the Hypertext Markup Language (HTML). In this version, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability. This specification is intended to replace (be a new version of) what was previously the HTML4, XHTML 1.0, and DOM2 HTML specifications.”

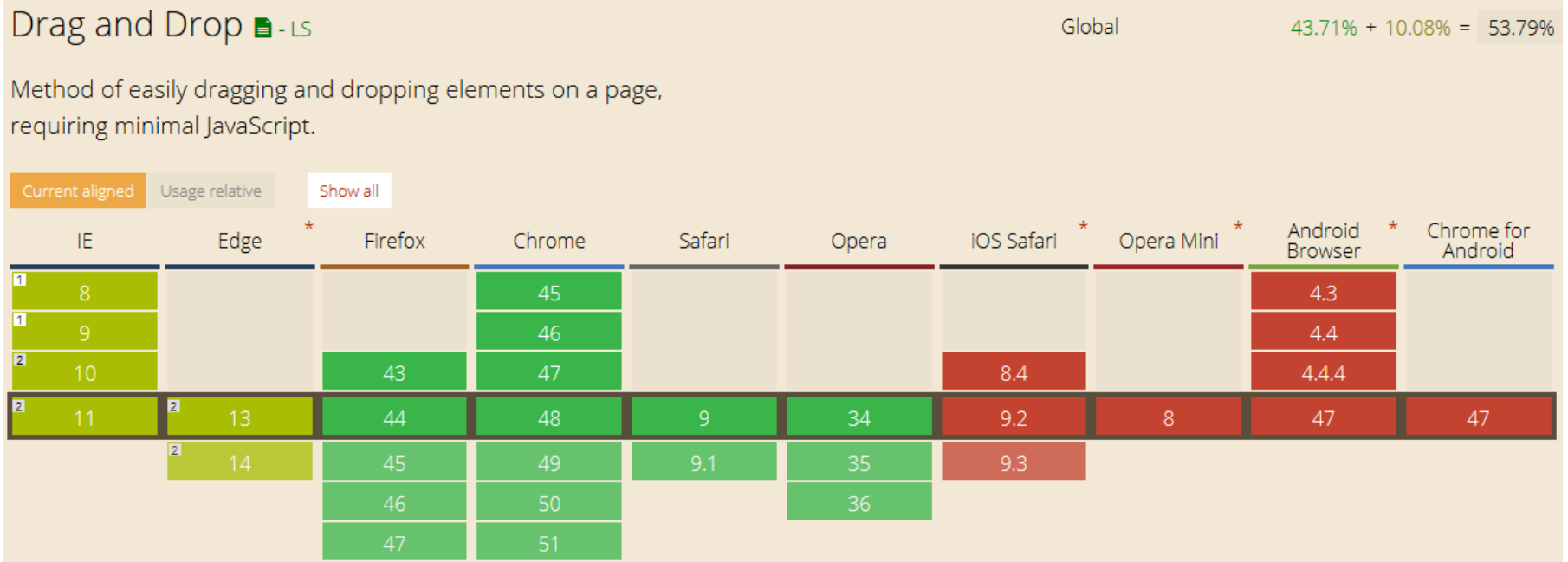
HTML5 overview

- Warning! HTML5 is a collection of features
 - Browsers give different support for each feature



Supported features

- Depends on browser
- Compatibility tables for every feature
 - E.g. <http://caniuse.com/>



HTML5 test

- How well does your browser support HTML5?
 - <http://html5test.com/>

YOUR BROWSER SCORES **478** OUT OF 555 POINTS



You are using Firefox 44.0 on Windows 7 Correct? ✓ ✕

Save results Compare to... Share Donate

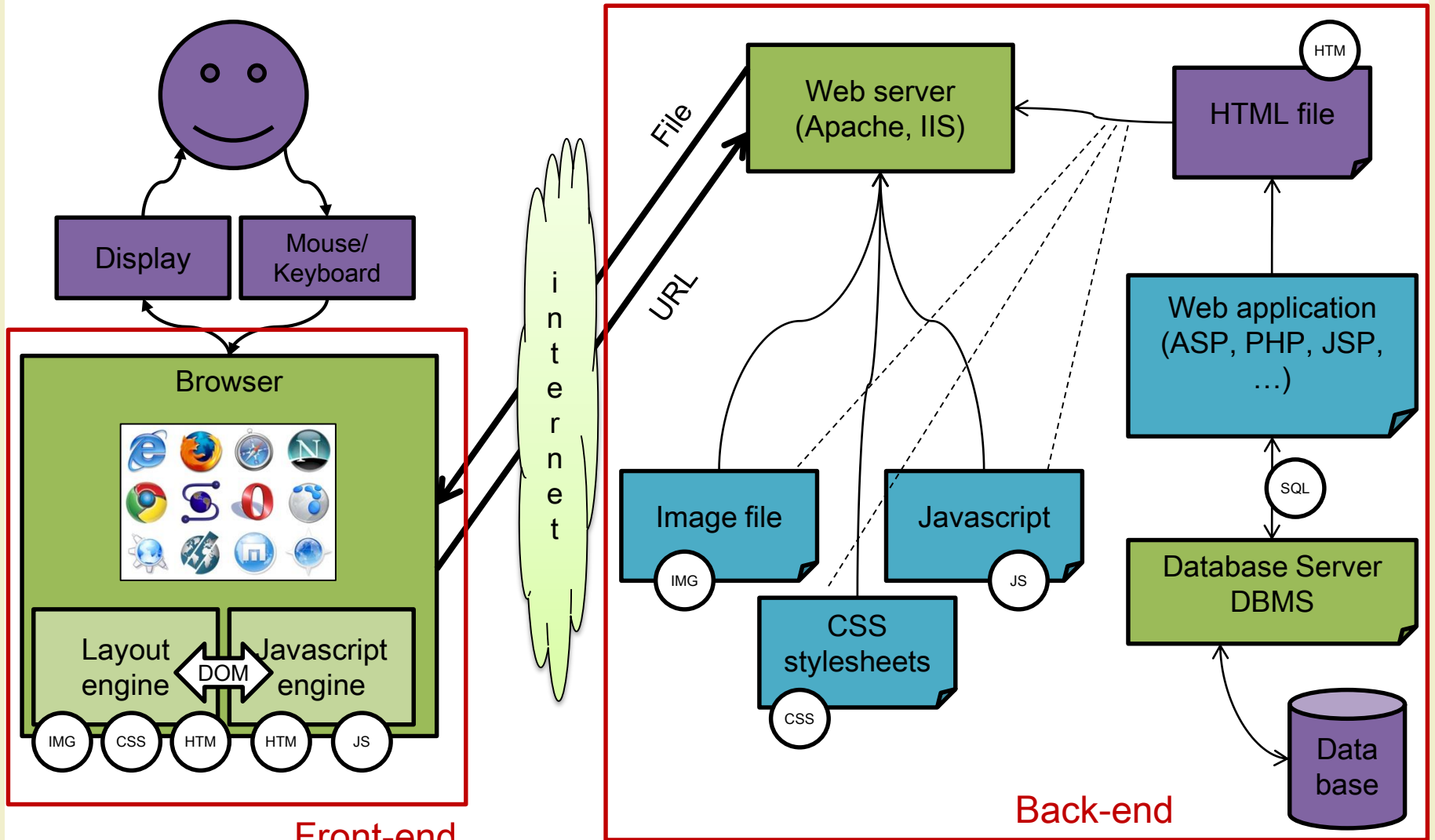
semantics	multimedia
Parsing rules 5	Video 31/35
<!DOCTYPE html> triggers standards mode Yes ✓	video element Yes ✓
HTML5 tokenizer Yes ✓	Subtitles Yes ✓
HTML5 tree building Yes ✓	Audio track selection No ✕
<i>HTML5 defines rules for embedding SVG and MathML inside a regular HTML document. The following tests only check if the browser is following the HTML5 parsing rules for inline SVG and MathML, not if the browser can actually understand and render it.</i>	Video track selection No ✕
Parsing inline SVG Yes ✓	Poster images Yes ✓
	Codec detection Yes ✓
	Advanced

Front-end development



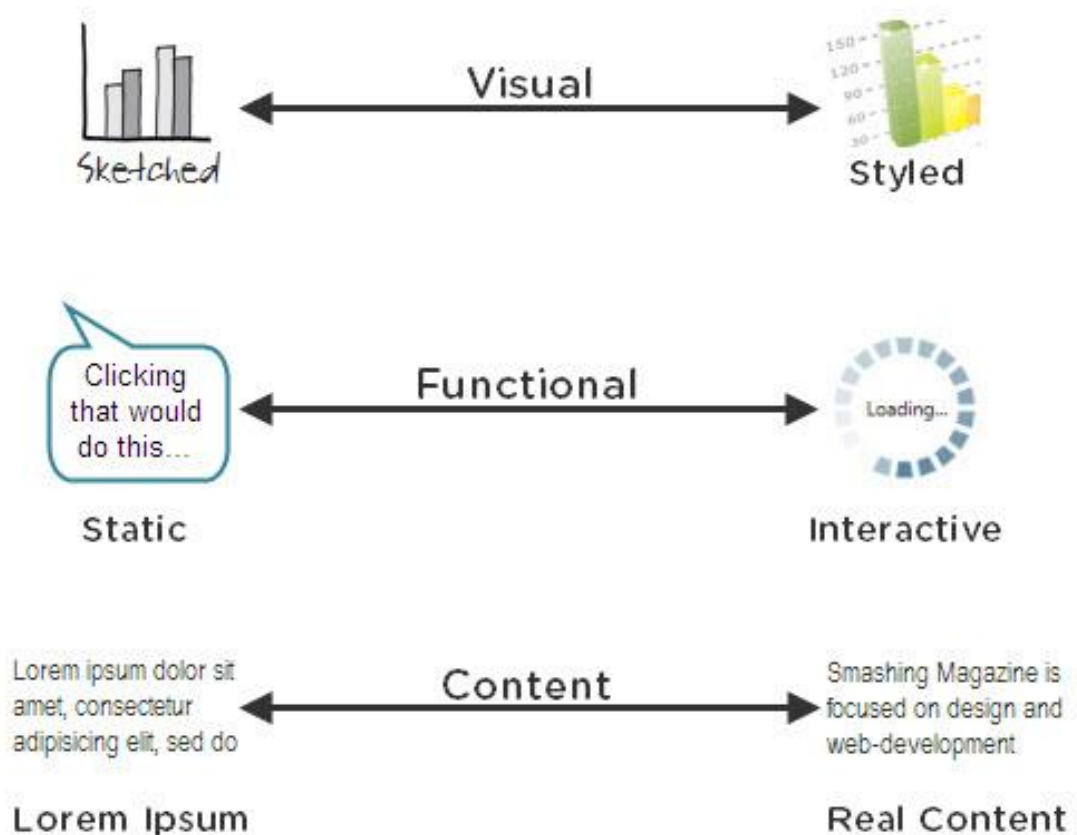
- Also known as client side development
- Practice of producing the front end of a website using
 - HTML5 (HyperText Markup language): the backbone of any website development process, that essentially structures the content of the website
 - CSS (Cascading Style Sheets): controls the presentation aspect of the site JavaScript
 - JavaScript (ECMAScript): an event-based language used to transform a static HTML page into a dynamic interface
 - Document Object Model (DOM), provided by the HTML standard, to manipulate a web page in response to events, like user input

General web architecture



Why front-end development?

- For rapid prototyping



HTML(5) basics

- HTML: HyperText Markup Language
- “De facto” standard
 - W3C: World Wide Web Consortium
 - <http://www.w3.org/>
- Continuous evolution
 - Born in 1991
 - HTML, HTML 2, HTML 4, XHTML 1, HTML 5
- Goal: to describe the structure of hypertext documents independently of computer platform
- Pure text: based on the first 127 characters of ASCII code

HTML(5) basics

- HTML describes a text by marking the various part that compose the document
 - Annotations use "tags"
- To visualize HTML documents: browser
 - Browsers interpret tags to visualize text in the proper way
 - The key to backwards compatibility: browsers ignore the tags they do not know
- To edit HTML documents: any text editor
 - Notepad, Notepad++, HAPedit, ...
 - We use WebStorm (JetBrains)

HTML5 document structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Sample page</title>
    <link rel="stylesheet" href="style-original.css" />
  </head>
  <body>
    ...
    ...
  </body>
</html>
```

Header

Body

- Header: useful information and links to other docs
- Body: anything in the browser window

Tags

- Each HTML tag describes different document content
- A tag is an expression between acute brackets (< >)
- Usually text portions are delimited by tag couples (e.g. <h1>Title</h1>)
- General rule: the final tag is identical to the initial one but starts with the / symbol
- Empty tags, i.e. that are not applied to a text, are written like this:

- Tags are the «elements» of an HTML5 document (we'll see the DOM – Document Object Model later)

Default display value

- Every HTML element has a default display value depending on what type of element it
- The default display value for most elements is block or inline
- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can)
 - Examples: `<div>`, `<h1>` - `<h6>`, `<p>`
- An inline element does not start on a new line and only takes up as much width as necessary
 - Examples: ``, `<a>`, ``

Attributes

- Tag can be better specified through attributes
- Example: specify that an image is aligned on the right or on the left, define the color of some text, specify the width of a table column, ...
- Attributes are variables with an assigned value
 - e.g. `width="100"`

HTML «traditional» tags and attributes

- Allow to
 - Write and format text and paragraphs
 - Write ordered and unordered lists
 - Define colors (fonts, backgrounds, links, ...)
 - Insert images
 - Insert hypertext links
 - Insert tables
 - Inserts forms
- HTML reference
 - <http://www.w3schools.com/tags/>
- HTML tutorial and examples
 - <http://www.w3schools.com/html/default.asp>

Frequently used tags

- To insert an image

```

```

– Images are stored in a separate file

- To insert a link

```
<a href=http://www.polito.it>Go to the homepage</a>
```

- To define paragraphs

```
<p>This is a paragraph</p>
```

- To define a block element in a document

```
<div>This is a block element</div>
```

- To define an inline section of a document

```
<span>This is an inline element</span>
```

What's new in HTML5

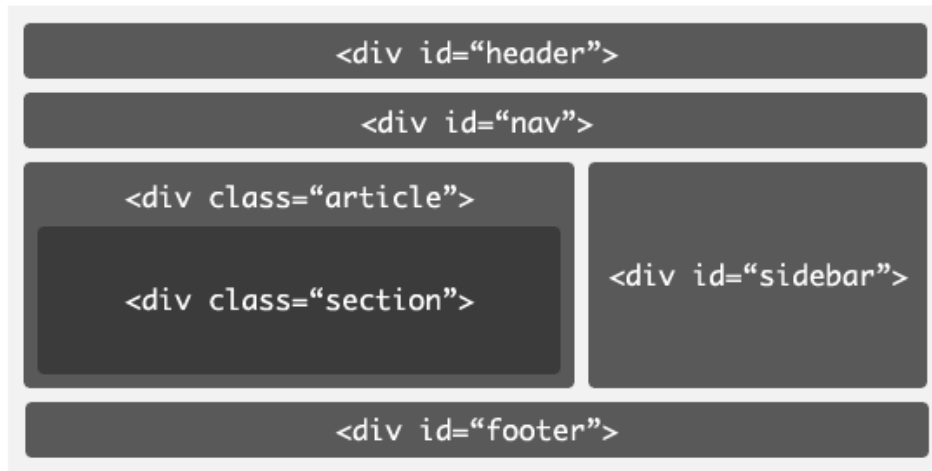
- New tags
- Much more



(Interesting) new tags

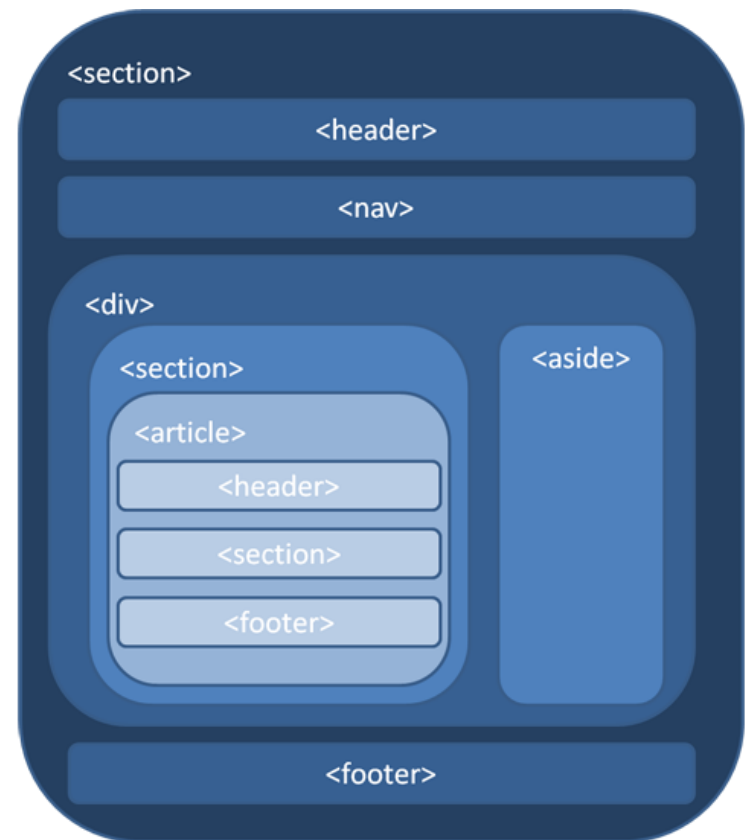
- Structural elements
 - Easier semantics for layout
- Forms
 - New input types
- Multimedia
 - Audio and video
 - Canvas (for drawing)

HTML5 layout tags



HTML4

HTML5



HTML5 layout tags

Tag	Meaning
<header>	Defines the top constant portion on all pages
<footer>	Defines the bottom constant portion on all pages
<section>	Primary text area of the current page
<article>	Defines topics/paragraphs that reside within a <section>
<aside>	Secondary section-like area with its own articles
<figure>	Area on the page that contains an image
<figcaption>	The caption that goes with the <figure>
<mark>	Highlights a part of the text
<hgroup>	Helps group the headings of a section
<time>	Helps define machine readable date/times

HTML5 layout tags

- CSS will do the magic!



Example

Foto.html



HTML5 media elements

- Inserting a video or an audio should be as easy as inserting an image
 - Browsers should have built-in support for playing video
 - No third-party plugins should be required
 - Standard formats should exist that are supported by all browsers
- HTML5 defines a standard way to embed video or audio in a web page, using a `<video>` or `<audio>` element



```
<image src="johann_sebastian_bach.jpg" />
```

HTML5 media elements

- New HTML5 media elements

Tag	Description
<code><audio></code>	For multimedia content, sounds, music or other audio streams
<code><video></code>	For video content, such as a movie clip or other video streams
<code><source></code>	For media resources for media elements, defined inside video or audio elements
<code><embed></code>	For embedded content, such as a plug-in

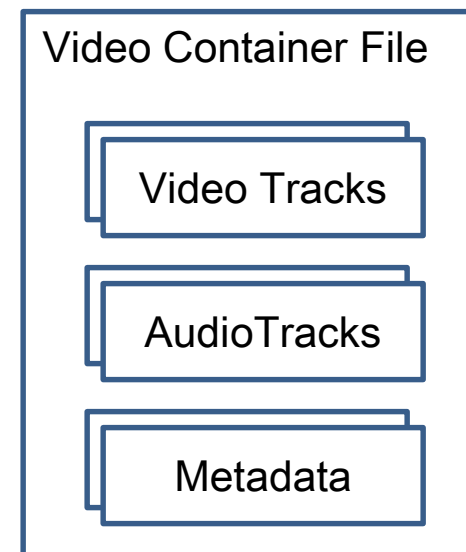
- The new audio and video tags make multimedia no longer a second-class citizen on the web
 - No separate download or enabled/disabled issues
 - No separate rendering (problems with HTML elements overlap)
 - Keyboard accessibility, styling with CSS, combining video and canvas

HTML5 media elements

- The media elements expose a common, integrated, and scriptable API to the document
 - You can design and program your own multimedia controls (e.g., play, seek, etc.)
- Examples
 - <http://www.craftymind.com/factory/html5video/CanvasVideo3D.html>
 - <http://www.craftymind.com/factory/html5video/CanvasVideo.html>

Audio and video files

- An audio or video file is just a container file, similar to a ZIP archive file that contains a number of files (audio tracks, video tracks, and additional metadata)
 - The audio and video tracks are combined at runtime to play the video
 - Metadata contains information about the video such as cover art, title and subtitle, captioning information, ...



Audio and video codecs

- Algorithms used to encode and decode a particular audio or video stream
- A codec is able to understand a specific container format and decodes the audio and video tracks that it contains
- Examples of audio codecs
 - MPEG-3: MPEG-1 or MPEG-2 Audio Layer III
 - AAC (Advanced Audio Coding): designed as the successor of the MP3 format and included in the MPEG-4 specification
 - Ogg Vorbis: open, patent-free, professional audio encoding and streaming technology from the Xiph.org Foundation
- Examples of video codecs
 - H.264: currently one of the most commonly used formats for the recording, compression, and distribution of high definition video
 - VP8: open video compression format released by Google
 - Ogg Theora: free and open video compression format from the Xiph.org Foundation

Audio and video codecs

- Some of the codecs are patented, others are freely available
 - For example, the Vorbis audio codec and the Theora video codec are freely available, while the use of the MPEG-4 and H.264 codecs are subject to license fees
- Originally, the HTML5 specification planned to require that certain codecs were supported
 - Unfortunately, there does not appear to be a single codec that all browser vendors are willing to implement
- For now, the codec requirement has been dropped from the specification
 - This decision might be revisited in the future...

Video formats

- Currently, there are 3 supported video formats for the video element
 - Ogg = Ogg files with Theora video codec and Vorbis audio codec
 - MPEG4 = MPEG 4 files with H.264 video codec and AAC audio codec
 - WebM = WebM files with VP8 video codec and Vorbis audio codec

<http://www.videojs.com/html5-video-support/>

Declaring a media element

audio.html

```
<!DOCTYPE html>
<html>
<title>HTML5 Audio </title>
<audio controls
  src="johann_sebastian_bach_air.ogg">
  An audio clip from Johann Sebastian Bach.
</audio>
</html>
```

- The controls attribute tells the browser to display common user controls for starting, stopping, and seeking in the media clip, as well as volume control



Declaring a media element

- Leaving out the controls attribute hides them, and leaves the clip with no way for the user to start playing
 - It will not show anything at all in the case of audio files, as the only visual representation of an audio element is its controls
- By including the autoplay attribute, the media file will play as soon as it is loaded, without any user interaction

```
<audio autoplay>  
  <source src="johann_sebastian_bach_air.ogg"  
    type="audio/ogg; codecs=vorbis">  
  An audio clip from Johann Sebastian Bach.  
</audio>
```

audio2.html

Multiple sources

- An alternate declaration can be used that includes multiple sources from which the browser can choose
- Sources are processed in order, so a browser that can play multiple listed source types will use the first one it encounters
- The beauty of this declaration model is that as you write code to interact with the media file, it doesn't matter to you which container or codec was actually used: the browser provides a unified interface for you to manipulate the media

```
<audio controls>  
  <source src="johann_sebastian_bach_air.ogg">  
  <source src="johann_sebastian_bach_air.mp3">  
  An audio clip from Johann Sebastian Bach.  
</audio>
```

HTML5 audio and video

- The HTML5 DOM has methods, properties, and events for the `<audio>` and `<video>` elements
- These methods, properties, and events allow you to manipulate `<audio>` and `<video>` elements using JavaScript
- Using HTML5 DOM, JavaScript and CSS together you can do interesting stuff
- Examples
 - audioCue.html
 - mouseoverVideo.html
 - videoCSS1.html, videoCSS2.html
 - videoJS.html
 - videoTimeline.html
 - videoCaption.html, videoCaption-lang.html
 - <http://chirls.com/2011/01/13/what-im-working-on-synchronized-videos-in-html5-featuring-ok-go/>

JS APIs for media control

Function	Behavior
<code>load()</code>	Loads the media file and prepares it for playback. Normally does not need to be called unless the element itself is dynamically created. Useful for loading in advance of actual playback.
<code>play()</code>	Loads (if necessary) and plays the media file. Plays from the beginning unless the media is already paused at another position.
<code>pause()</code>	Pauses playback if currently active.
<code>canPlayType(type)</code>	Tests to see whether the video element can play a hypothetical file of the given MIME type.

Media attributes

Read-only attribute	Value
<code>duration</code>	The duration of the full media clip, in seconds. If the full duration is not known, <code>NaN</code> is returned.
<code>paused</code>	Returns <code>true</code> if the media clip is currently paused. Defaults to <code>true</code> if the clip has not started playing.
<code>ended</code>	Returns <code>true</code> if the media clip has finished playing.
<code>startTime</code>	Returns the earliest possible value for playback start time. This will usually be <code>0.0</code> unless the media clip is streamed and earlier content has left the buffer.
<code>error</code>	An error code, if an error has occurred.
<code>currentSrc</code>	Returns the string representing the file that is currently being displayed or loaded. This will match the source element selected by the browser.

Media attributes

Attribute	Value
<code>autoplay</code>	Sets the media clip to play upon creation or query whether it is set to <code>autoplay</code> .
<code>loop</code>	Returns <code>true</code> if the clip will restart upon ending or sets the clip to loop (or not loop).
<code>currentTime</code>	Returns the current time in seconds that has elapsed since the beginning of the playback. Sets <code>currentTime</code> to seek to a specific position in the clip playback.
<code>controls</code>	Shows or hides the user controls, or queries whether they are currently visible.
<code>volume</code>	Sets the audio volume to a relative value between 0.0 and 1.0, or queries the value of the same.
<code>muted</code>	Mutes or unmutes the audio, or determines the current mute state.
<code>autobuffer</code>	Tells the player whether or not to attempt to load the media file before playback is initiated. If the media is set for auto-playback, this attribute is ignored.

Additional video attributes

Attribute	Value
<code>poster</code>	The URL of an image file used to represent the video content before it has loaded. Think “movie poster.” This attribute can be read or altered to change the poster.
<code>width</code> , <code>height</code>	Read or set the visual display size. This may cause centering, letterboxing, or pillaring if the set width does not match the size of the video itself.
<code>videoWidth</code> , <code>videoHeight</code>	Return the intrinsic or natural width and height of the video. They cannot be set.

HTML5 forms

- Formerly called Web Forms 2.0
- Native functionality (no scripting for validation)
 - Means less coding
- New input types
 - Date and color pickers
 - Search, e-mail, web address input types
 - Validation
 - Spin boxes and sliders
- Backward compatible
 - Features degrade gracefully (unknown input types are treated as text-type) input

Form creation

- Form tag with a few attributes
 - Name: form name
 - Action: URI (resource) that will do data processing
 - Method: method for passing parameters from the form to the destination URI ("POST" or "GET" or "PUT")
- A form contains several input elements

```
<form name = "datiUtenti" action = "URI" method = "POST" >  
  Elementi di input  
</form>
```

Form example

- Input elements
 - Text field
 - Checkbox
 - Radio button
 - “Submit” button
 - “Reset” button
 - ...
 - Text
 - Images

	Articolo	Immagine	Taglia	Quantità	Prezzo (taglia medium)
<input checked="" type="checkbox"/>	Maglia girocollo arancio		small	<input type="text" value="1"/>	61.00 €
<input type="checkbox"/>	Maglia dolcevita blu		medium	<input type="text" value="0"/>	70.20 €
<input checked="" type="checkbox"/>	Camicia righe azzurre		medium small medium large	<input type="text" value="3"/>	25.00 €
<input checked="" type="checkbox"/>	Tuta ginnastica		large	<input type="text" value="2"/>	45.70 €
<input type="checkbox"/>	Pantalone velluto grigio		medium	<input type="text" value="0"/>	53.50 €

Modalità di pagamento:

- Contanti alla consegna
- Tessera prepagata
- Carta di credito (2.50 € di commissione)

Invia l'ordine

Annulla



Input elements

- General structure (with a few exceptions)

```
<input type="text" name="indirizzo" size="30"  
       value="Inserisci qui il tuo indirizzo" />
```

- Tag "input" with some attributes
 - Type: type of the element
 - Name: name of the element
 - Value: the valued that the form passes to the destination URI
 - Other attributes specific to the element type (e.g. size for input type "text")

Input elements example

<input checked="" type="checkbox"/>	Camicia righe azzurre		<select><option>medium</option><option>small</option><option>medium</option><option>large</option></select>	<input type="text" value="3"/>	25.00 €
					

```
<input type="checkbox" name="art3" value="1" />
```

```
<input type="image" name="camicia" src="./img/camicia_righe.jpg" height="80"/>
```

```
<select name="dim3">  
  <option value="1">small</option>  
  <option value="2" selected>medium</option>  
  <option value="3">large</option>  
</select>
```

```
<input type="text" name="q3" value="0" size="2" />
```

Input elements example

Modalità di pagamento:

- Contanti alla consegna
- Tessera prepagata
- Carta di credito (2.50 € di commissione)

Element selected during page loading

```
<input type="radio" name="pag" value="0" checked />Contanti alla consegna<br />  
<input type="radio" name="pag" value="1" />Tessera prepagata<br />  
<input type="radio" name="pag" value="2" />Carta di credito  
(2.50 € di commissione)
```

Important: same name

Invia l'ordine

Annulla

```
<input type="submit" name="invia" value="Invia l'ordine" />  
<input type="reset" name="annulla" value="Annulla" />
```

Input elements example

- Password fields: shows bullets or stars instead of characters

- `input type="password"`

```
<input type="password" maxlength="8" size="18" name="passwd" />
```

- Textarea field

- `textarea`

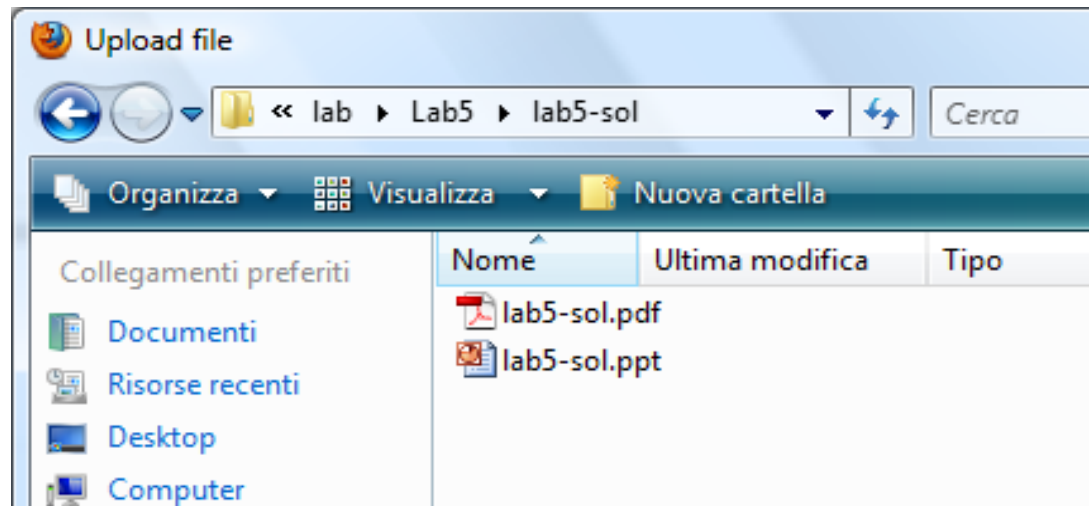
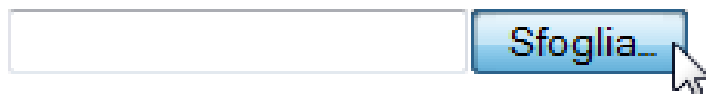
Qui puoi scrivere il tuo testo

```
<textarea name="testo" rows="5" cols="40">
  Qui puoi scrivere il tuo testo
</textarea>
```

Input elements example

- File
 - input type="file"

```
<input name="fileUtente" type="file" size="20" />
```



New input types

- Allow for better input control and validation
- If not supported, they will behave as regular text fields

<http://wufoo.com/html5/>

Browser Support for New HTML5 Input Types

							
	Firefox	Safari	Safari	Chrome	Opera	IE	Android
Email ?	4+	5+	3.1+	6+/10+	10.6+	10+	4+
Tel ?	4+	5+	3.1+	6+	10.6+	10+	2.3+
Url ?	4+	5+	3.1+	6+/10+	10.6+	10+	2.3+
Search ?	4+	5+	4+	6+	10.6+	9/10+	4+
Color ?	29+	8+	8-	20+	11+	11-	4.4+
Number ?	29+	5+	3.2+	7+	9+	10+	2.3+
Range ?	23+	4+	5+	6+	11+	10+	4.2+
Date ?	32-	7-	5+	20+	9+	11-	4.4+
Text ?	All	All	All	All	All	All	All

email input type

- The value of the email field is automatically validated when the form is submitted

E-mail:

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

  <form action="demo_form.asp" method="get">
    E-mail: <input type="email" name="user_email" /><br />
    <input type="submit" />
  </form>

</body>
</html>
```

url input type

- The value of the url field is automatically validated when the form is submitted

Homepage:

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

<form action="demo_form.asp" method="get">
  Homepage: <input type="url" name="user_url" /> <br />
  <input type="submit" />
</form>

</body>
</html>
```

number input type

- Restrictions on numbers

Points:

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

  <form action="demo_form.asp" method="get">
    Points: <input type="number" name="points" min="1"
      max="10" />
    <input type="submit" />
  </form>

</body>
</html>
```

Restrictions on input type number

Attribute	Value	Description
max	<i>number</i>	Specifies the maximum value allowed
min	<i>number</i>	Specifies the minimum value allowed
step	<i>number</i>	Specifies legal number intervals (if step="3", legal numbers could be -3,0,3,6, etc)
value	<i>number</i>	Specifies the default value

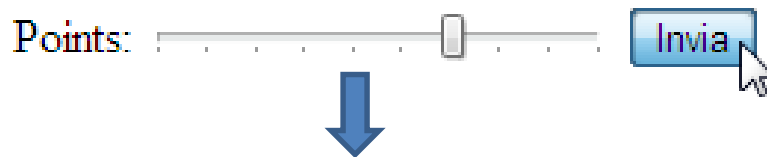
```
<!DOCTYPE HTML>
<!DOCTYPE HTML>
<html>
<body>
  <form action="demo_form.asp" method="get">
    <input type="number" name="points" min="0" max="10"
      step="3" value="6" />
    <input type="submit" />
  </form>
</body>
</html>
```

range input type

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

  <form action="demo_form.asp" method="get">
    Points: <input type="range" name="points"
      min="1" max="10" />
    <input type="submit" />
  </form>

</body>
</html>
```



Input was received as:

points=7

Date pickers

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

  <form action="demo_form.asp" method="get">
    Date: <input type="date" name="user_date" />
    <input type="submit" />
  </form>

</body>
</html>
```

Date: 2011-01-21 ▾

Gennaio							2011
Lun	Mar	Merc	Gio	Ven	Sab	Dom	
27	28	29	30	31	1	2	
3	4	5	6	7	8	9	
10	11	12	13	14	15	16	
17	18	19	20	21	22	23	
24	25	26	27	28	29	30	
31	1	2	3	4	5	6	

Date pickers

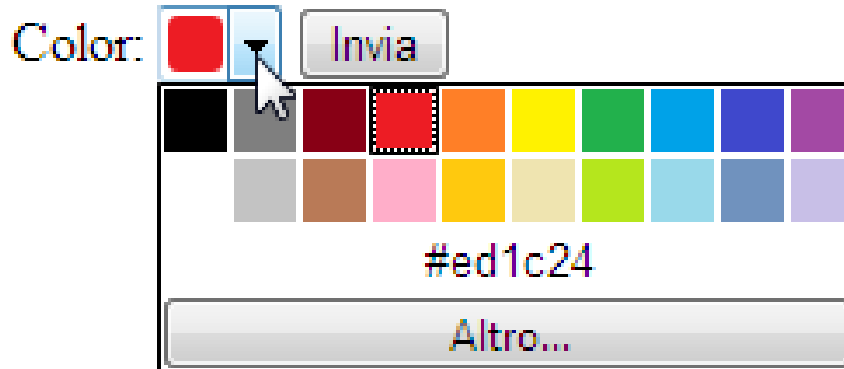
- New input types for selecting date and time
 - date - selects date, month and year
 - month - selects month and year
 - week - selects week and year
 - time - selects time (hour and minute)
 - datetime - selects time, date, month and year (UTC time)
 - datetime-local - selects time, date, month and year (local time)

color input type

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

  <form action="demo_form.asp" method="get">
    Color: <input type="color" name="user_color" />
    <input type="submit" />
  </form>








</body>
</html>
```



New form attributes

- Some of them:

<http://wufoo.com/html5/>

Browser Support for New HTML5 Input Attributes								
		Firefox	Safari	Safari	Chrome	Opera	IE	Android
Placeholder	?	4+	4+	4+	10+	11.10+	10+	2.3+
Autofocus	?	4+	5+	5-	6+	11+	10+	3+
Maxlength	?	4.4+	5+	4+	6+	11+	9/10	2.3+
List (Datalist)	?	4+	7-	7-	20+	9+	10+	4.3-
Autocomplete	?	4+	5.2+	6+	14+	10.6+	11+	4.4+
Required	?	6+	5+	4+	6+	10.6+	10+	2.3+
Pattern	?	4+	5+	4+	10+	11+	10+	2.3+
Spellcheck	?	3.6+	4+	7+	10+	11+	10+	4.3-

pattern attribute

- Specifies a pattern used to validate an input field
- A pattern is a regular expression (or “regexp”)

Country code:

```
<!DOCTYPE HTML>
<html>
<body>
  <form action="demo_form.asp" method="get">
    Country code: <input type="text" name="country_code"
      pattern="[A-z]{3}" title="Three letter country code" />
    <input type="submit" />
  </form>
</body>
</html>
```

Regular expressions

```
^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$
```

- **^[a-zA-Z0-9._-]+**
 - The email address must begin with alpha-numeric characters (both lowercase and uppercase characters are allowed): it may have periods, underscores and hyphens
- **@**
 - There must be a '@' symbol after initial characters
- **[a-zA-Z0-9.-]+**
 - After the '@' sign there must be some alpha-numeric characters; it can also contain period and and hyphens
- **\.**
 - After the second group of characters there must be a period ('.'); this is to separate domain and subdomain names.
- **[a-zA-Z]{2,4}\$**
 - Finally, the email address must end with two to four alphabets; {2,4} indicates the minimum and maximum number of characters

Examples of e-mail patterns

```
[a-zA-Z0-9!#$%&'*/=?^_`{|}~-]+  
(?:\.[a-zA-Z0-9!#$%&'*/=?^_`{|}~-]+)*  
@(?:[a-zA-Z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+  
[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
```

```
[a-zA-Z0-9!#$%&'*/=?^_`{|}~-]+  
(?:\.[a-zA-Z0-9!#$%&'*/=?^_`{|}~-]+)*  
@(?:[a-zA-Z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+  
(?:[a-zA-Z]{2}|com|org|net|edu|gov|mil|biz|  
info|mobi|name|aero|asia|jobs|museum)\b
```

HTML5 is much more

- But you need to know JavaScript and DOM first
- Later on (possibly...)
 - More on audio and video, synchronization
 - Canvas & animations
 - Drag&drop
 - Offline web applications
 - Touch events
 - History
 - ...

References

- HTML5 specifications
 - <https://www.w3.org/TR/html5/>

License

- This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
- You are free:
 - to Share - to copy, distribute and transmit the work
 - to Remix - to adapt the work
- Under the following conditions:
 - Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
 - Noncommercial - You may not use this work for commercial purposes.
 - Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit <http://creativecommons.org/license/by-nc-sa/3.0/>