**Course outline**

**Module 0: Course information**

- Welcome to "JavaScript Introduction"  
- Course outline, grading and certificate  
- Course forums  
- Course tools

**About W3C and the Web**

- About W3C and the Web  
- Why accessibility is important  
- Why internationalization is important

**Module 1: Introduction to JavaScript**

1.1 Introduction to Module 1  
1.2 JavaScript, HTML and CSS  
1.3 JavaScript overview  
1.4 Your first HTML/CSS/JS page  
1.5 Variables, values, functions, operators and expressions  
1.6 Simple JavaScript examples to play with  
1.7 Exercises - Module 1

**Module 2: Adding interactivity to HTML**

2.1 Introduction to Module 2  
2.2 Conditional statements, loops and logical operators  
2.3 Functions and callbacks  
2.4 Handling events  
2.5 The DOM API  
2.6 Let's write a small game  
2.7 Exercises - Module 2

**Module 3: Playing with HTML5 APIs**

3.1 Introduction to Module 3  
3.2 Arrays and iterators  
3.3 HTML5 multimedia and JavaScript API  
3.4 Displaying a map with the Geolocation API  
3.5 Playing sound samples and music  
3.6 Exercises - Module 3

**Module 4: Structuring data**

4.1 Introduction to Module 4  
4.2 Objects: properties and methods  
4.3 Creating multiple objects  
4.4 Improving the game with classes  
4.5 Exercises - Module 4

**Module 5: Working with forms**

5.1 Introduction to Module 5  
5.2 Built-in JavaScript objects  
5.3 HTML5 tables, forms and input fields  
5.4 The JSON notation  
5.5 A small application  
5.6 Further reading  
5.7 Final exam and conclusion

## Online editor tools

 Bookmarked

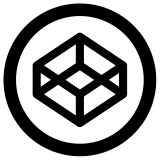
### Online editor tools

To help you practice during the whole duration of the course, you will use the following online editor tools. Pretty much all the course's examples will actually use these.

#### JsBin

[JsBin](http://jsbin.com/) is an open source collaborative Web development debugging tool. This tool is really simple, just open the link to the provided examples, look at the code, look at the result, etc. And you can modify the examples as you like, you can also modify / clone / save / share them.  
  
Tutorials can be found on the Web (such as <http://code.tutsplus.com/tutorials/javascript-tools-of-the-trade-jsbin--net-36843>) or on YouTube.  Keep in mind that it's always better to be logged in (it's free) if you do not want to lose your contributions/personal work.

#### CodePen

[CodePen](http://codepen.io/) is an HTML, CSS, and JavaScript code editor that previews/showcases your code bits in your browser. It helps with cross-device testing, real-time remote pair programming and teaching.

This is a great service to get you started quickly as it doesn't require you to download anything and you can access it, along with your saved projects from any Web browser. Here's an article which will be of-interest if you use CodePen: [10 Cool Things You Can Do with CodePen and JavaScript](https://www.sitepoint.com/cool-things-codepen-javascript/) [Chris Coyier, June 13, 2016].

There are many other handy tools such as [JSFiddle](http://jsfiddle.net/" \t "_blank), and [Dabblet](http://dabblet.com/" \t "_blank) (Lea Verou's tool that we will use extensively in a future CSS course). Please share your favorite tool on the discussion forum, and explain why! Share also your own code contributions, such as a nice canvas animation, a great looking HTML5 form, etc. Sharing them using JS Bin, or similar tools, would be really appreciated.

#### JavaScript debuggers

Here is a selection of tools to help debug JavaScript code. The instructor will indicate other tools in module 1 of the course.

* [Firefox JS debugger](https://developer.mozilla.org/en-US/docs/Tools/Debugger) (debugger shipped inside Firefox) - all other browsers have integrated debuggers as well.
* [JS Lint](http://www.jslint.com/)  - The JavaScript Code Quality Tool
* [CodeBeautify](http://codebeautify.org/jsvalidate)  - JavaScript Validator

## W3C cheatsheet

 Bookmarked

### The W3C cheatsheet

The [W3C cheatsheet](http://www.w3.org/2009/cheatsheet/) provides quick access to useful information from a variety of specifications published by W3C. It aims at giving in a very compact and mobile-friendly format a compilation of useful knowledge extracted from W3C specifications, completed by summaries of guidelines developed at W3C, in particular Web accessibility guidelines, the Mobile Web Best Practices, and a number of internationalization tips.

Its main feature is a lookup search box, where one can start typing a keyword and get a list of matching properties/elements/attributes/functions in the above-mentioned specifications, and further details on those when selecting the one of interest.

The W3C cheatsheet is no longer available as an Android app, but remains available as a [pure Web application](http://dev.w3.org/2009/cheatsheet/doc/).

## First steps in Web accessibility

 Bookmarked

### First steps in Web accessibility

There are many simple Web accessibility improvements that you can implement and check right away, even when you are new to this topic. Two example excerpts are provided below on this page but you can find more tips and information from W3C/WAI:

* [Tips for Getting Started with Web Accessibility](https://www.w3.org/WAI/gettingstarted/tips/)
* [Easy Checks - A First Review of Web Accessibility](https://www.w3.org/WAI/eval/preliminary)

#### Example 1: page title

Good page titles are particularly important for orientation — to help people know where they are and move between pages open in their browser. The first thing screen readers say when the user goes to a different Web page is the page title. In the Web page markup, they are the <title> within the <head>.

**Check #1: There is a title that adequately and briefly describes the content of a page, and that it distinguishes the page from other Web pages.**

Example:

1. <head>
2. ...
3. <title>Web Accessibility Initiative (WAI) - home page</title>
4. ...
5. </head>

#### Example 2: image text alternatives ("ALT TEXT")

Text alternatives ("alt text") are a primary way of making visual information accessible, because they can be rendered through any sensory modality (for example, visual, auditory or tactile) to match the needs of the user. Providing text alternatives allows the information to be rendered in a variety of ways by a variety of user agents. For example, a person who cannot see a picture can have the text alternative read aloud using synthesized speech.

**Check #2: Every image has alt with appropriate alternative text.**

Example: See the W3C logo below. It contains a link that points to the W3C Web site. The text alternative is going to be a brief description of the link target.

[W3C web site](http://w3.org/)

1. <a href="http://w3.org">
2. <img src="images/w3c\_home.png" width="72" height="48" alt="W3C Web site">
3. </a>

## What is Web internationalization?

 Bookmarked

# What is internationalization?

Access to the Web for all has been a fundamental concern and goal of the W3C since the beginning. It is easy to overlook the needs of people from cultures different to your own, or who use different languages or writing systems, but you have to ensure that any content or application that you design or develop is ready to support the international features that they will need.

'Internationalization' is sometimes abbreviated to 'i18n' in English, because there are 18 characters between the 'i' and the 'n'.

The [W3C Internationalization Activity](http://www.w3.org/International/) works with W3C working groups and liaises with other organizations to make it possible to use Web technologies with different languages, scripts, and cultures.

During this course you will learn about some basic internationalisation features, such as character encoding and language declarations. If you don't use those features you will create barriers for people from different cultures who are trying to access your content. This is important even if you think you are only designing for a specific community – communities are made up of diverse individuals, and the Web stretches worldwide.

## Unicode

Text in a computer or on the Web is composed of characters. Characters represent letters of the alphabet, punctuation, or other symbols.

**Unicode is a universal character set, ie. a standard that defines, in one place, all the characters needed for writing languages in use on computers. It is a superset of all other character sets that have been encoded.**

As a content author or developer, you should nowadays always [choose the UTF-8 character encoding](http://www.w3.org/International/questions/qa-choosing-encodings) for your content or data. This Unicode encoding is a good choice because you can use a single encoding to handle any character you are likely to meet. This greatly simplifies things.

## Essential steps in Web internationalization

 Bookmark this page

# Essential steps in Web internationalization

You find below three examples (and checks!) to help you to ensure that your Web page works for people around the world, and to make it work differently for different cultures, where needed. Let's meet the words 'charset' and 'lang', soon to become your favorite markup ;)

## Example 1: Character encoding declaration

A character encoding declaration is vital to ensure that the text in your page is recognized by browsers around the world, and not garbled. You will learn more about what this is, and how to use it as you work through the course.  For now, just ensure that it's always there.

**Check #1: There is a character encoding declaration near the start of your source code, and  its value is UTF-8.**

Example 1:

1. <head>
2. <meta charset="utf-8"/>
3. ...
4. </head>

## Example 2: Primary language declaration

For a wide variety of reasons, it's important for a browser to know what language your page is written in, including font selection, text-to-speech conversion, spell-checking, hyphenation and automated line breaking, text transforms, automated translation, and more. You should always indicate the primary language of your page in the <html> tag. Again you will learn how to do this during the course.  You will also learn how to change the language, where necessary, for parts of your document that are in a different language.

**Check #2: The HTML tag has a lang attribute which correctly indicates the language of your content.**

Example 2: This indicates that the page is in French.

1. <!doctype html>
2. <html lang="fr">
3. <head>
4. ...

## Example 3: Cultural bias

People around the world don't always understand cultural references that you are familiar with, for example the concept of a 'home run' in baseball, or a particular type of food. You should be careful when using examples to illustrate ideas. Also, people in other cultures don't necessarily identify with pictures that you would recognize, for example, hand gestures can have quite unexpected meanings in other parts of the world, and photos of people in a group may not be representative of populations elsewhere.  When creating forms for capturing personal details, you will quickly find that your assumptions about how personal names and addresses work are very different from those of people from other cultures.

**Check #3: If your content will be seen by people from diverse cultures, check that your cultural references will be recognized and that there is no inappropriate cultural bias.**

## Don't worry!

The following 7 quick tips summarize some important concepts of international Web design. They will become more meaningful as you work through the course, so come back and review this page at the end.

* 1. **Encoding**: use the UTF-8 (Unicode) character encoding for content, databases, etc. Always declare the encoding.
  2. **Language**: declare the language of documents and indicate internal language changes.
  3. **Navigation**: on each page include clearly visible navigation to localized pages or sites, using the target language.
  4. **Escapes**: use characters rather than escapes (e.g. &#xE1; &#225; or &aacute;) whenever you can.
  5. **Forms**: use UTF-8 on both form and server. Support local formats of names/addresses, times/dates, etc.
  6. **Localizable styling**: use CSS styling for the presentational aspects of your page. So that it's easy to adapt content to suit the typographic needs of the audience, keep a clear separation between styling and semantic content, and don't use 'presentational' markup.
  7. **Images, animations & examples**: if your content will be seen by people from diverse cultures, check for translatability and inappropriate cultural bias.

You will find more quick tips on the [Internationalization quick tips](http://www.w3.org/International/quicktips/) page. Remember that these tips do not constitute complete guidelines.

## Internationalization checker

When you start creating Web pages, you can also run them through the W3C's [Internationalization Checker](https://validator.w3.org/i18n-checker/).  If there are internationalization problems with your page, this checker explains what they are and what to do about it.

## HTML is for structure

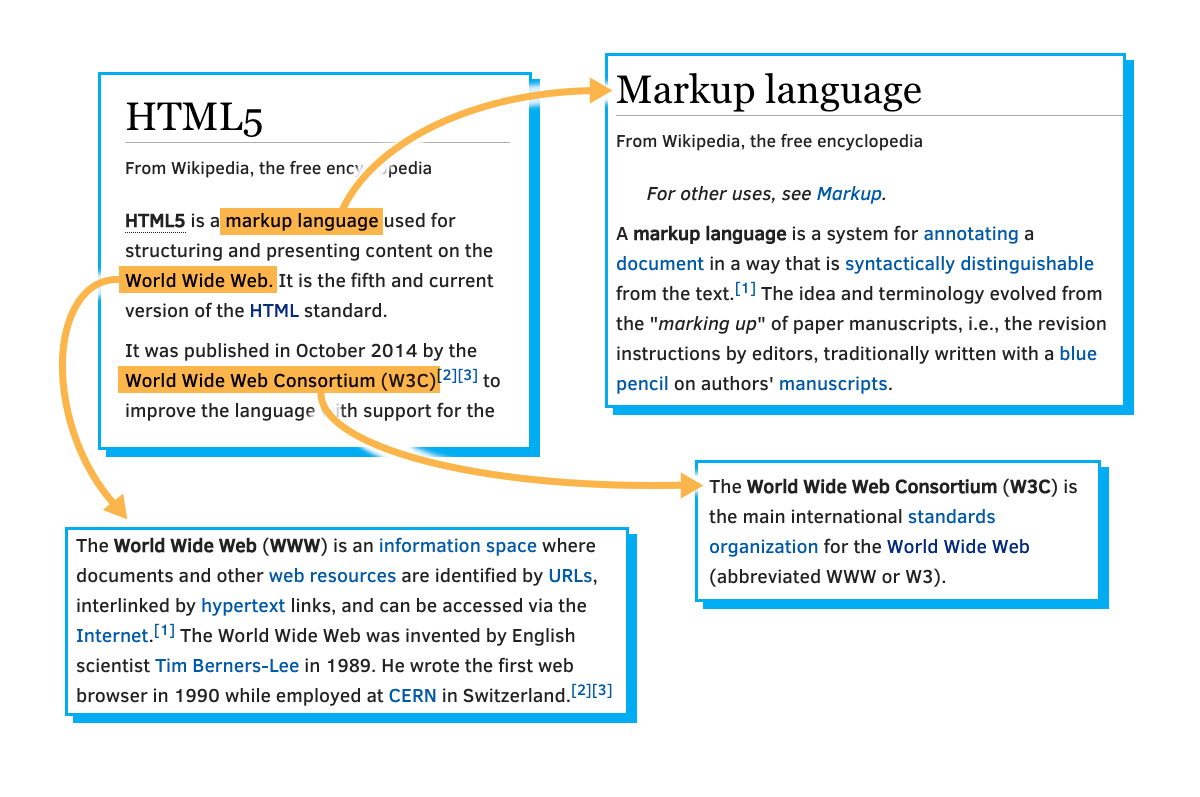
 Bookmark this page

### HTML5 logoHTML: Hyper Text Markup language

#### The "****H****yper ****T****ext" part: links!

A fundamental key to the World Wide Web is the concept of "hypertext".  Hypertext is built on the idea of linking information together, not unlike using footnotes, but far easier and more flexible. The idea is to "mark up" your document with links and define how to break it down into different segments (chapters, sections, paragraphs, tables, figures, etc.)

That's why, in 1989, Tim Berners-Lee began to create a definition of HTML: Hypertext Markup Language, to provide a simple, uniform way to incorporate hyperlinks into a text document.



He envisioned a technology that would facilitate thoroughly interconnected documents. He wanted authors to be able to connect an idea in one document to the source of the idea in another, or connect a statement with the data that backs up that statement. Traditionally this kind of thing was done with footnotes and bibliographies, which can be cumbersome. This information should be easily transferable from one place to another, so that in reading one document, it is easy to access everything related (linked) to it. Tim Berners-Lee imagined a "Web" of interconnected documents.

He used the metaphor of a Web to emphasize the importance of connections between documents. It was not just a long list of details, but rather a sea of information stretching out in all directions. This sea of information was navigated by a new tool called a "browser".

#### The "Markup" part : elements, tags and attributes!

So the "M" in HTML stands for "Markup", but what does Markup really mean?  Essentially it means to annotate a document with extra information: things like where different sections and paragraphs begin and end, which part is the title, which things should be emphasized and so on.

There are many ways to markup a document, but HTML borrows a technique from an ancestor language, SGML ([Standard Generalized Markup Language](https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language)), which uses angle brackets ("<" and ">") to separate the annotations from the regular text.  In HTML these annotations are called "tags".

For example, consider the following chunk of HTML code

1. <body>
2. <h1>A Tale of Two Cities</h1>
3. <p>
4. It was the best of times, it was the worst of times, . . . .
5. </p>
6. . . .
7. <p>
8. . . . it is a far, far better rest
9. that I go to than I have ever known.
10. </p>
11. </body>

If you eliminated everything in between the angle brackets from the text, for most purposes it would still read the same:

A Tale of Two Cities  
It was the best of times, it was the worst of times . . . .  
 . . .  
. . . it is a far, far better rest  
 that I go to than I have ever known.

Once you know that everything in angle brackets is "meta-information", it gives you a lot of flexibility. You can put a lot of different things in between those brackets without any of it showing up (directly) in your finished document. And though you don't usually see directly what's in those angle brackets, they can often have a big effect on how your Web page looks, as well as how it responds and interacts with you.

**HTML Elements**

If you are sitting at a coffee shop next to a table of Web developers, you will probably hear three words quite a bit: "Tags", "Attributes" and "Elements".

"Elements" are the pieces themselves, i.e. a paragraph is an element, a header is an element, even the body is an element. Most elements can contain other elements, as the body element would contain header elements, paragraph elements, in fact pretty much all of the visible elements of the Document Object Model (that developers refer to as the "DOM").

As an example, let's look at a simplified version of the last HTML code we showed you:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <title>Your first HTML page</title>
5. <meta charset="utf-8"/>
6. </head>
7. <body>
8. <h1>My home page</h1>
9. <p>Hi! Welcome to my Home Page! My name is Michel Buffa, I'm a professor at the University of Nice, in France, and I'm also the author of two MOOCS about HTML5 on W3Cx.</p>
10. </body>
11. </html>

Click the red circle next to HTML to unfold this HTML document structure (we can also say "see its DOM structure"):

Consider the figure above. It contains a single html element. It turns out this includes within it the entire content of your html file. If you click on the "html" red node, you'll find that it contains two components, a head and a body. Clicking on each of those will reveal their respective contents. This structure is what we computer scientists call a "tree". Any given element (except for the outermost "html" element) is wholly contained inside another element, referred to as the "parent" element. Not surprisingly, the elements contained within a given element are its "child" elements. And, yes, children of a common parent are often referred to as "siblings".

Thus in the example above, the top element is the html element, which contains just two elements, the head and body.  The head element contains a title element and the body contains an h1 element and a p element.  In a more typical example, the body would contain many more children, but for our purpose this is enough. p is for "paragraph" (the text between <p> and </p> will be separated by some space before the next element is displayed in the final HTML page rendering), h1 means "heading level 1", and will be rendered by default in bold with a bigger char size than any other text element, etc.

That may be a great picture, but how do we represent such a structure in a text file?  Well, that's where "tags" come in.

#### Tags



# <html>

### <BODY>

#### <p>

##### <em>

"Tags" are what we use to organize a text file (which is just a long string of characters) such that it represents a tree of elements that make up the html document. Tags are not the elements themselves, rather they're the bits of text you use to tell the computer where an element begins and ends. When you "mark up" a document, you generally don't want those extra notes that are not really part of the text to be presented to the reader.

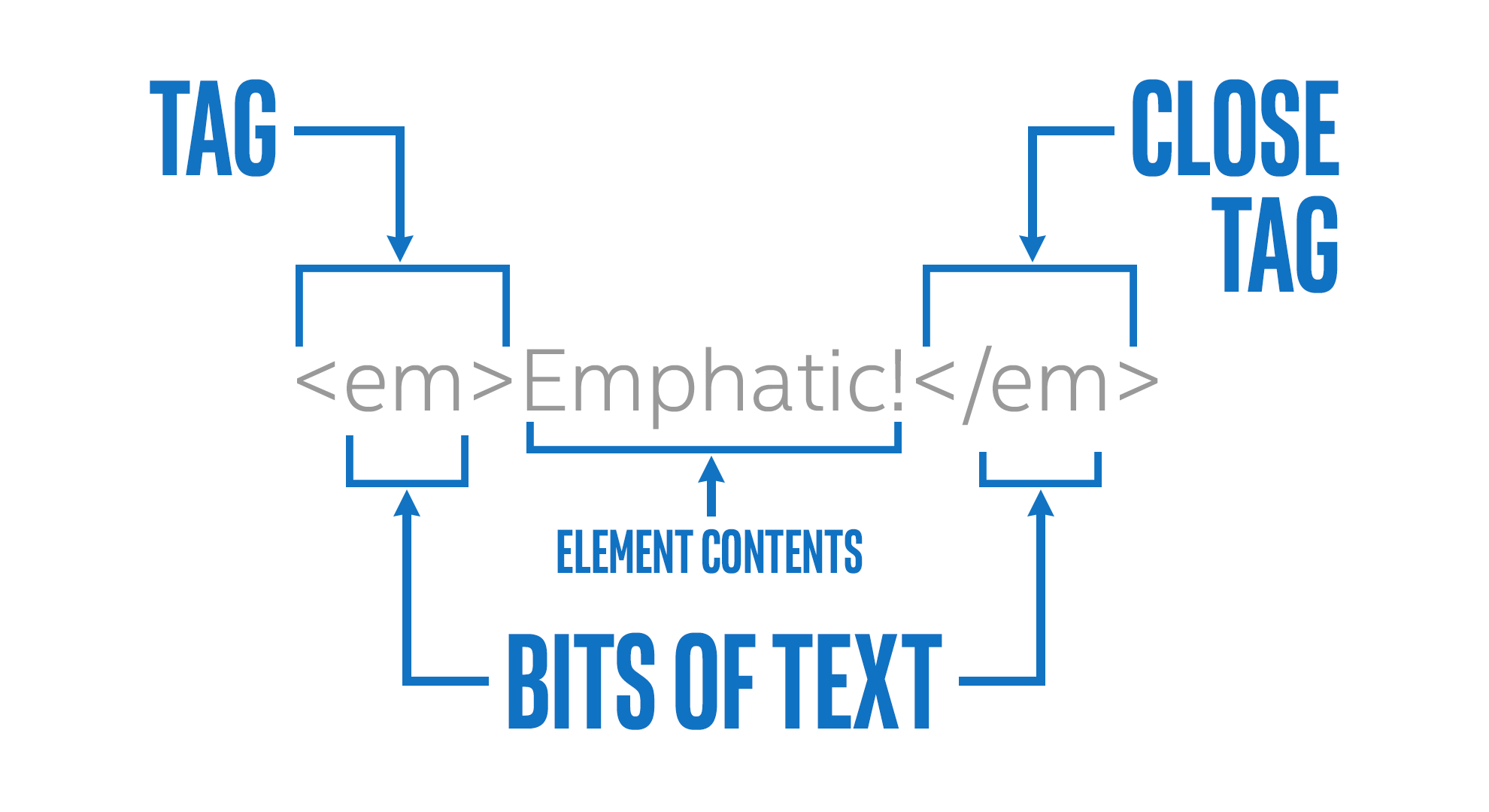
HTML borrows a technique from another language, SGML, to provide an easy way for a computer to determine which parts are "MarkUp" and which parts are the content. By using "<" and ">" as a kind of parentheses, HTML can indicate the beginning and end of a tag, i.e. the presence of "<" tells the browser "this next bit is markup, pay attention".

Whatever that tag (or "open tag") does, it applies to the content following the tag. Unless you want that to be the entire rest of the document, you need to indicate when to stop using that tag and do something else, so "<" and ">" are used again. Since elements are typically nested within other elements, the browser needs to be able to distinguish between the end of the current tag and the beginning of a new tag (representing a nested element). This is done by adding a "/" right after the "<" to indicated that it's a "close tag". To indicate the beginning and end of a paragraph (indicated by the single letter "p") you end up with something like this:

1. <p>This is my first paragraph!</p>

The browser sees the letters "<p>" and decides "A new paragraph is starting, I'd better start a new line and maybe indent it". Then when it sees "</p>" it knows that the paragraph it was working on is finished, so it should break the line there before going on to whatever is next.

For example, the "<em>" tag is used for element that needs Emphasis.  The  "<" and ">" indicate that this is a tag, and the "little bits of text" in between tell us what kind of tag it is.  To completely describe the element, it needs an open and close tag, and everything in between the tags is the contents of the element:



Most tags have open and close versions, but there are a few strange ones.  For more info, we strongly recommend that you follow the W3Cx [HTML5&CSS Fundamentals](https://www.edx.org/course/html5-introduction-w3cx-html5-0x-0) course, but we generally refer to the strange ones as "self closing" tags.   Usually these tags represent an element that is completely described by its attributes, and thus there is no need for other content.  So if you see something like this:

1. <img src="https://goo.gl/pVxY0e" alt="Floating Flower"/>

... then you should know that the slash at the end of the open tag is sort of a shorthand for a close tag, so you won't see any other indication that this element is now complete. There are also a few tags that don't even use the "/" at the end, they just don't have any close tag at all.  This works because all of the information this tag needs is declared in an "attribute".

The <img> tag is one of them, the "/" at the end is optional and can be removed entirely, this will still be [valid HTML5](http://w3c.github.io/html/syntax.html#void-elements).

1. <img src="https://goo.gl/pVxY0e" alt="Floating Flower">

These elements, without a "/" at the end, are called "void elements". They are : area, base, br, col, embed, hr, img, input, link, menuitem, meta, param, source, track, wbr.

#### Attributes

Most of what you can learn about HTML attributes is presented [in the three W3Cx MOOCs about HTML5](https://www.edx.org/school/w3cx) (fundamentals, coding essentials, and advanced techniques), but we can introduce the idea briefly in this JavaScript course. Basically, a given element on your Web page can be distinguished by any number of unique or common attributes. For example, we've already seen how an image can be inserted in your Web page, and in that example we used the width attribute of the <img> tag in order to constrain the width of the image:

1. <img src="https://pbs.twimg.com/profile\_images/110455194/n666194627\_2302\_400x400.jpg"
2. width=200 alt="Michel Buffa plays rock&roll">

As you might guess, the <img> tag also has a height attribute, as well as others. Different HTML tags share some common attributes that we'll meet in the next section, which are particularly useful when coupled with CSS (id and class) for applying graphic styles (color, shadow, etc.), but  can also have specific attributes (for example: the src attribute can be found in the <video>, <audio>, <img> tags but not on a <p> or on an <h1> tag!)

**CSS is for style**

 Bookmarked

**Definition**

CSS, or **C**ascading **S**tyle **S**heets, is a style sheet language used to describe the way an HTML or XML document should look to a user. CSS is where you specify the color, size, spacing, font and other visual aspects of the content that you create in your markup language document.

Usually you will see CSS used alongside HTML to describe the way a Web page looks and feels. You can have a Web page without CSS, but it would be very difficult to make it look the way you want with just HTML. This is why almost every Web page is a combination of HTML and CSS.

**CSS** • /si-ɛs-ɛs/ • *noun*

Stands for "**C**ascading **S**tyle **S**heets". A style sheet language for describing how to display an HTML document.

**Let's look at an example!**

Let's take the (ridiculous) Michel Buffa home page again:

Notice the use of some HTML tags: h1, img, p, body etc.

Now, we can add some "CSS rules" to the HTML, and see that the appearance of the resulting HTML page rendering is rather different (click on the HMTL/CSS buttons to see alternatively the HTML or the CSS code, remember you can always make changes to the code: change the color in the CSS part, etc.):

If you click on the CSS button on the top left of the previous codepen example, you will see the CSS rules that have been applied to the HTML document. Let's look at the first one:

1. h1 {
2. color:red;
3. background-color:lightGreen;
4. border:12px solid violet;
5. padding: 5px;
6. border-radius: 15px;
7. text-align: center;
8. }

This rule turns all the h1s in the document into red text, centered horizontally, on a light green background, with a violet border of 12 pixels (a solid border, not a dashed one), and this border has rounded corners made of arcs of a circle whose radius is 15 pixels.

The part before the opening brace (line 1) is the "CSS selector", it indicates the elements that will have their properties changed according to what is inside the braces.

The part inside the braces is a set of properties and values that will be useful for setting the look and feel of the selected elements.

Line 2 for example, says that all h1s will be colored in red.

**CSS rules are applied in sequence**

After the previous rule is applied, then the second rule is taken into account, then the next, etc. In this way, you can see that all h2s will be brown (second rule).

The third rule uses what is called "a multiple selector":

1. p, h1, h2 {
2. font-family: cursive
3. }

This one says that all p, h1 and h2 will use a cursive font character. The "," means "and also".

This is also how we indicate in the last rule that images and paragraphs should be moved to the right 50 pixels (property margin-right: 50px)

**The id and class attributes**

Basically, any given element on your Web page can be identified uniquely with an 'id' attribute, or grouped with a class of other elements by setting the 'class' attribute.

1. <p id="paragraph-1" class="regular-paragraphs">
2. Call me Ishmael . . .
3. </p>

The paragraph above has a unique identifier: the id attribute whose value is "paragraph-1" and is part of a class of "regular-paragraphs". The letters inside the quotes have no meaning to the computer, they just need to be consistent. They are actually strings.

Again, the fact that the computer does not care what we put in those strings (except for some restrictions) means we can use them to convey meaning to a human developer. I could just as easily have said id='x' and class='y', but anyone looking at that would have no hint what the significance of x and y are. Best practice is to name these things to increase clarity, consistency and brevity.

Let's look at a modified version of the Michel Buffa's home page example:

The last two rules first target the element whose id is 'hobbyTitle', in our case it's the second h2 element:

1. <h2 **id="hobbyTitle"**>My Hobbies</h2>

And here is the CSS rule:

1. **#hobbyTitle** {
2. font-family: 'caveat';
3. font-size:40px;
4. text-shadow: 4px 4px 2px rgba(150, 150, 150, 1);
5. }

Line 1 uses the "#" character in the selector, meaning that we're going to select an element by its id attribute. In this case, the selector equal to #hobbyTitle, will select the element that has an attribute id="hobbyTitle".

In that case we use a funny char font called 'caveat' we took from the Google font service (see [fonts.google.com](https://fonts.google.com/)), and in order to be able to use it in a font-family CSS property, we included its definition using a <link> tag in the HTML part of the document:

1. <head>
2. <title>Your first HTML page</title>
3. <meta charset="utf-8"/>
4. **<link href="https://fonts.googleapis.com/css?family=Caveat"**
5. **rel="stylesheet">**
6. </head>

The last rule targets all elements that have an attribute class="funny". Notice they can be different elements, we can have a p and an h3element that have the class="funny" attribute:

1. **.funny**{
2. color:purple;
3. font-family: 'caveat';
4. font-size:40px;
5. }

This rule will change the color, font family and size of two out of three paragraphs in the HTML element:

1. ...
2. <p class="funny">I also play electric guitar and love coding WebAudio applications...</p>
3. ...
4. <p class="funny">Music, Movies, Video Games, Traveling, Family, etc.</p>

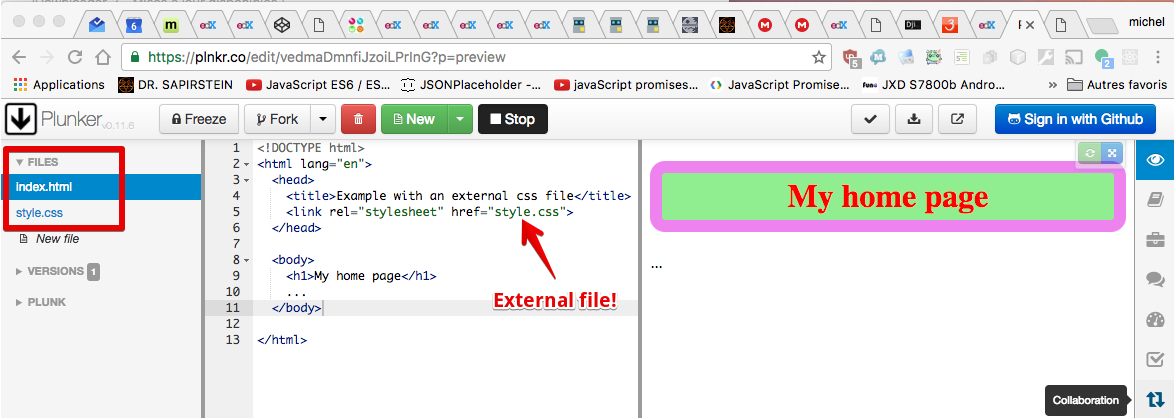
There are many, many, many different CSS properties in existence, and many different ways to select elements. We recommend that you follow the W3Cx [CSS Basics and HTML5&CSS Fundamentals](https://www.edx.org/school/w3cx) courses to learn more about CSS and about HTML5 basics.

**Where can we put the CSS rules: In the HTML file? In another file?**

You can do both!

You can embed the CSS rules between a <style>...</style> tag, located inside the <head>...</head> of the HTML documents, like in this example:

This is OK if you do not have too many CSS rules. In general it's better to put the CSS rules in one or more separate .css files, like this ([open this example in Plunker](https://plnkr.co/edit/vedmaDmnfiJzoiLPrInG?p=preview))



Note that when you use an online IDE, you usually type/paste the CSS rules in a "CSS tab" in the online editor, and it will hide all the plumbery for you (except the more complete ones such as [plunker](https://plnkr.co/) or [c9.io](https://c9.io/) that will enable you to manage files in the cloud).

### JavaScript is the interactive glue between HTML and CSS

JavaScript is the third of the "magic trio": HTML5/CSS/JavaScript. It is the only programming language a browser can run (without installing any plugins or extensions), and it's a real standard of the Web (even if not standardized by the W3C).

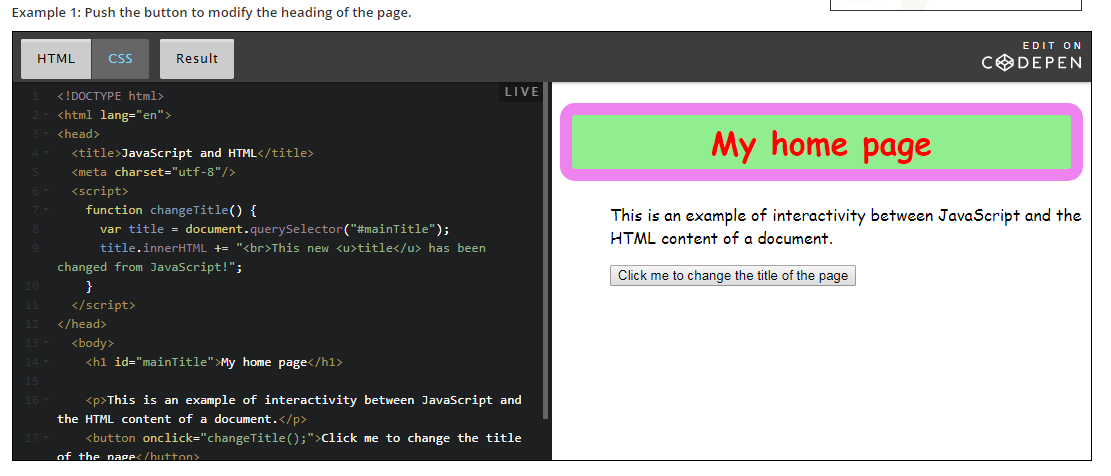
#### Why do we call it "the interactive glue between HTML and CSS"?



Actually, this description does not do justice to JavaScript, which can do far more than just act as glue. JavaScript can be run outside of the browser (on a nodeJS interpreter on a remote server, or in scripts run by the operating system), but for this intro course, we will focus on "JavaScript" in the browser (an advanced course about "server side JavaScript" is on its way at W3Cx).

So, in the browser, JavaScript lies between HTML and CSS and will be used together with these two languages. Let's take a look at two small examples:

##### Example 1: Push the button to modify the heading of the page.



##### Example 2: Push another button to modify the CSS style (color, background-color, border) of a paragraph in the page.

We will take a detailed look at how these examples work in the subsequent parts of the course. These examples are just here to show you how JavaScript can interact with the HTML content and the CSS styles of a Web document.

Notice that in these examples, the JavaScript code is located in the HTML of the document.