

Working with the Web — A tutorial for understanding how websites get made.

VI.0

By Angus A. A. Mol (Leiden University Centre for Digital Humanities)



This tutorial is available under a Creative Commons [Attribution-ShareAlike 4.0 International](https://creativecommons.org/licenses/by-sa/4.0/) license.

What you will need for this tutorial:

- An internet connection
- A browser (this tutorial uses Google's Chrome browser as an example)
- Notepad++ or a similar text editor

Introduction

This tutorial was designed to give you an understanding of how websites get made, also giving you some handholds to start making your own websites.

In Part 1, we will look at what the Web even is and how it works. It also tells you what the first steps are in getting a website up and running.

Part 2 will take you back to the 2nd millennium, when the internet was young and we were all playing around with its basic building blocks.

Part 3 transports us back to the now and will provide an interactive demo and some hand's on working with Wordpress CMS, which powers about a third of the contemporary WWW.

This tutorial is based on my experiences creating websites since the late nineties for personal use, and recently for use as part of the digital humanities minor. It also builds on my experiences coordinating website development for my previous employer, the [Prince Claus Fund](#).

Some additional tips:

- The first time **key concepts** are introduced, they will be bolded.
- Any hyperlinks are in blue and underlined — they will work in the PDF, but they won't work on paper ;-). This first hyperlink will take you to the start of the tutorial at <http://workweb.lucdh.nl/tutorial>.
- Any time you need to take an action, it is underlined! Navigate to the start of the tutorial now.
- Arrows → are used to indicate next steps in a workflow (e.g. File → Save)
- Italics means whatever text is in *italics*, should be visible on your screen.

What are these blocks?

This is where I will add additional information, which we may skip due to time constraints during the tutorial, but which may be fun or interesting to read back later.

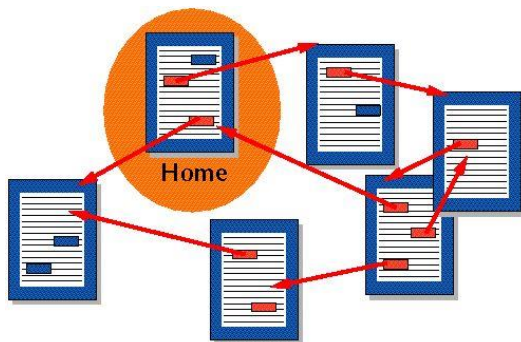
As this is the first time running this tutorial, apologies for any mistakes and typos in it. In addition, I'd be grateful if you have any feedback for me.

Part I, The basic building blocks of websites

Step I.1: How does the Web even work?

At its core, the **World Wide Web** (WWW) is a large and ever-growing global collection of documents and other **web resources**, i.e. things that can be accessed via the web. The vast majority of the web resources you will interact with on a daily basis are **hypertext** web pages.

Hypertexts are documents that have the ability to **link** and be linked by other documents, allowing you to quickly switch between them.



1. A schematic representation of hypertexts and the links between them.

For example, go to the home section of the webpage that is part of this tutorial (<https://webwork.lucdh.nl/tutorial>) and navigate to Step I.1.

While the ability to navigate from page to page may seem ‘basic stuff’ to us now, it was one of the pillars that allowed for the creation of the World Wide Web in 1989.

Other milestones that came together to form the World Wide Web were:

- **Hypertext Markup Language**, more on this below!
- **Browsers**, i.e. software to access web resources, function as a web **clients** or web **user agent**.
- Hypertext Transfer Protocol (**HTTP**). A protocol is a formal and agreed upon way of doing something, in the case of the HTTP is concerns how computers communicate with each other on the WWW. A client, for example a browser user agent (itself operated by a human user), will make a *request* for information with a **server**. A **server** is a machine that quite literally serves documents and other information, storing it and returning it when asked. For example, servers can host a web page. The request to a server returns a *response*.

The Internet is the same thing right?

The term WWW or Web is often used interchangeable with the “Internet”. Strictly speaking, this is wrong: the Internet is the network of computers and other devices connected to each other and all the telecommunication hardware that is needed to connect them. A way to think about it is that, where the Internet is the ‘medium’, the WWW is the ‘message’.

Where is the first webpage?

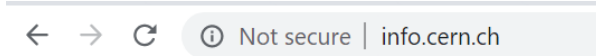
The World Wide Web was invented by Tim Berners-Lee at [CERN](https://www.cern.ch) and the web was first made publicly available in 1990. The very first webpage was actually lost. In the beginning most webpages were stored on hard disks (so not saved on always online servers). The optical storage drive containing the very first webpages were lost at a conference in California in the early nineties. CERN has endeavoured to reconstruct it at and this can be seen at : <http://info.cern.ch/> .

Webpages get lost all the time, due to a variety of human and technical reasons: the life of the average webpage is only about 100 days.

There exist many variants of client-server protocols, but in 1990 HTTP was a new and very efficient way of communicating with machines connected to the Internet, and was set as the standard for the WWW.

Check the full name of the webpage for this tutorial: you will see it, as well as all other web resources starts with 'http'.

Of course, updates were made to HTTP since then, including Hypertext Transfer Protocol Secure (**HTTPS**), which allows for encrypted communications between client and server. HTTPS is very important for secure communication and has become a new standard on the web. Google's Chrome browser, for example, will mark any web page that does not make use of HTTPS as 'Not Secure'.



Step 1.2: Web addresses

Likely one of the ways, aside from clicking on hyperlinks, you most frequently start communication between a client and a server is by typing in a web address in the address bar of the browser. This **URL** or Unique Resource Locator, is quite literally the unique name of the location of a document or other web resource. Not one URL will be the same. The reason for this is that underneath this URL 'hides' a IP, an Internet Protocol address. An IP, together with HTTP, is the 'actual' numeric address of the machine that you are trying to connect with. Every single entity connected to the internet has its own IP address.

You can see your current public IP address, by typing in "IP address" in a google search.

URLs were a later addition to the WWW, because it turned out it was prohibitively difficult for people to remember and type in numeric addresses to connect to web pages.

In short, most web pages will have their own url in the form of a domain name and getting ownership of a certain **domain name** is a very first important step for creating any sort of website. You may build a beautiful place online, but if no one knows at what address it can be found, it will not receive any visitors.

Any domain registration is contractually binding and will incur costs, so we can't practice all steps involved in getting ownership of a domain in this tutorial.

You can still have some fun thinking of domain names and seeing if they are still available for your desired domain extension, using popular domain registrars: for example, go to <https://domains.google/> and see if your domain name of choice is still available.

Who decides on all these things?

There is actually a consortium that functions as a *de facto* governing body for the Web: the World Wide Web Consortium or [W3C](#). Members are governmental, commercial, or academic organisations, who jointly decide and innovate on the many standards and protocols that keep the Web working properly.

Something similar applies to the registration of domain names, which is overseen at the top level by the Internet Corporation for Assigned Names and Numbers ([ICANN](#)).

What makes a good domain name?

Coming up with domain names is very hard, as most ones that can easily be thought of have been registered already.

Ideally: keep it short (no phrases, not more than three words), but descriptive (unless an acronym or abbreviation is very well known among your target audience), and simple (no numbers, no hyphens, etc.).

The domain name ending, or extension, top-level domains (.com, .org, .nl, .io, etc.) is not important from a technical point of view and has more to do with national identity or branding (.com has global recognition, .tv used to be hot in the naughties, .io is the extension for hip new initiatives).

If you go through the entire process of domain name registration, the domain will become linked to a legal entity (a person or organization) for as long as the contract between you and your domain registrar is valid. This is important to understand, as it is possible for [anyone looking up a domain](#) via a WHOIS service, to see who is the owner of that domain (unless you take steps during domain registration to have your identity hidden from WHOIS lookups).

Step 1.3: Hosting

Once you have registered a domain to you or the organisation you represent, you need to find a place that holds the web resources you want to share with the world.

Basically any computer can be a server, but the problem with most computers is that they are not very reliable for communication purposes. For example, you could use your laptop as a server, but anytime you would disconnect from the internet or simply put it in sleep mode, your website would no longer be available for anyone to visit. A golden standard for servers is to be **up for 99,99% of the time** (i.e. only being inaccessible for only 52.60 minutes per year).

Also, as servers are always online, they can be **vulnerable to attacks**. In short, you need to know what you are doing when hosting a server. That is why the vast majority of people and organizations that own websites will have a contract with an external party specializing in hosting websites.

There will be costs associated with this, frequently an annual fee. Picking a good hosting party can be a bit of a minefield as there are frequently a lot of small letters and **hidden costs**. One thing to look out for is what will happen when traffic (visitors to) your website will (suddenly) increase, total amount of storage you have for resources (documents, images, videos, etc, i.e. just like the amount of storage on your computer's hard drive).

Another important part is what type of server you rent and the services that come with this. Generally speaking all major hosting companies should provide you with an easy way to configure your websites, such as by using **CPanel**. You will find more on the use of Cpanel in Part 3.

Recapping, these are the three basic building blocks for any website:

- **Content**, in the form of hypertext documents and other web resources
- **Address**, in the form of a url, acquired through domain name registration.
- **Hosting**, in the form of an 'always' online, secure server, most often acquired via a contract with a hosting party.

Part I of this tutorial has only provided a basic overview of some pretty important and complex aspects of developing and running a website. As there are costs and legal aspects (of ownership and liability) associated certainly do inform yourself if you want to own, build, and develop any sort of website. If you work together with a partner for web development, many if not all of these steps will be handled for you. but it is quite important that you understand these basics as it will allow you to make the right decisions for you or the organisation you are representing.

Part II, HyperText Markup Language (HTML)

Step 2.1 What is HTML?

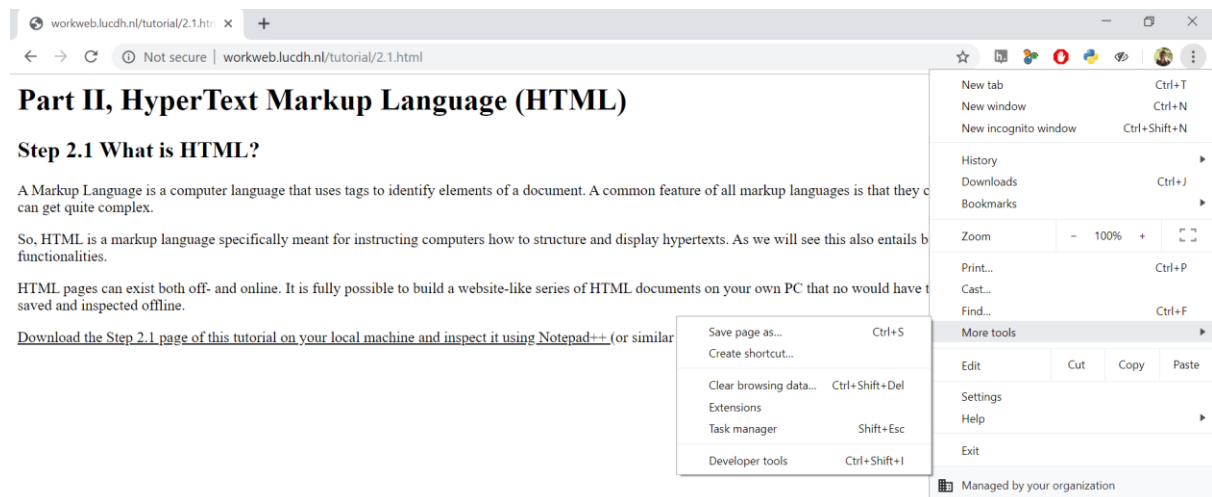
A Markup Language is a computer language that uses tags to identify elements of a document. A common feature of all markup languages is that they can be read (and written) by humans, even if it can get quite complex.

So, HTML is a markup language specifically meant for instructing computers how to structure and display hypertexts. As we will see this also entails basic layouting capabilities as well as other functionalities.

HTML pages can exist both off- and online. It is fully possible to build a website-like series of HTML documents on your own PC that no would have to see online. In addition, webpages can be saved and inspected offline.

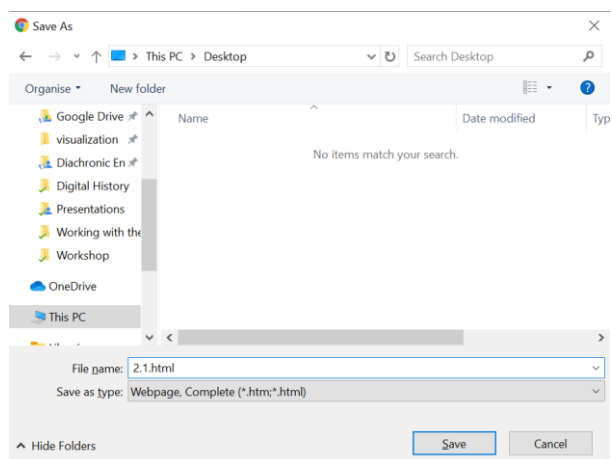
Download the Step 2.1 page of this tutorial on your local machine and inspect it using Notepad++ (or similar text editor software).

Example using Chrome:

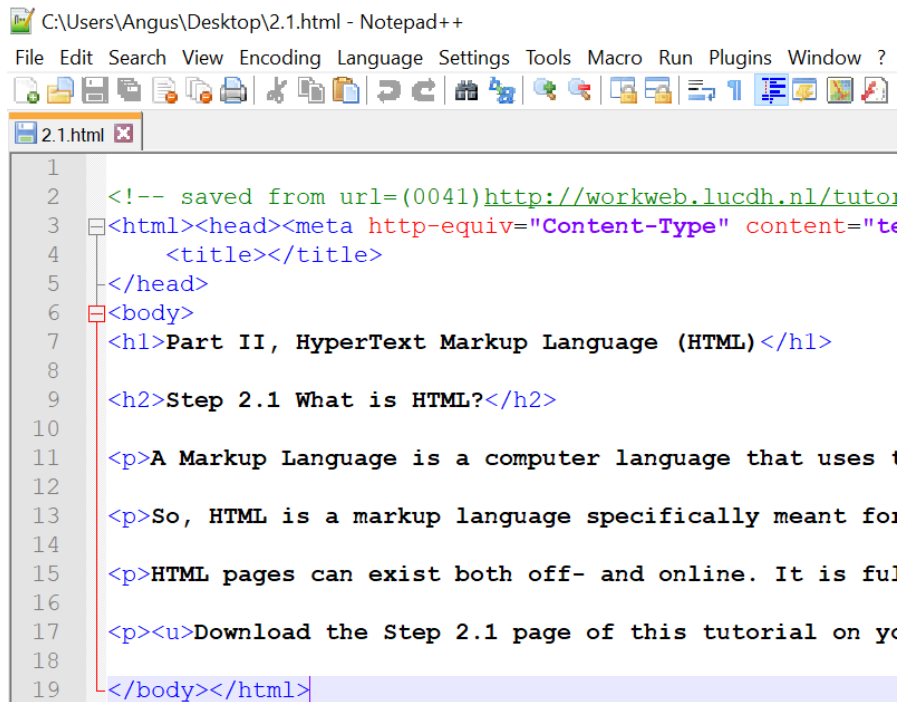


- ➔ Go to the Chrome button (the three dots in the top right corner), select More tools, Save page as. Alternatively you can simply use the shortcut Ctrl+S.
- ➔ Save the .html file somewhere on your local machine, e.g. your desktop.
- ➔ Open it with Notepad++ (either right click on the file → "open with Notepad++" or open Notepad++ and use Open → File and navigate to the file.

What you see now is the 'raw' html view of the webpage you downloaded.



Step 2.2 HTML Tags



```
1
2 <!-- saved from url=(0041)http://workweb.lucdh.nl/tuto
3 <html><head><meta http-equiv="Content-Type" content="te
4 <title></title>
5 </head>
6 <body>
7 <h1>Part II, HyperText Markup Language (HTML)</h1>
8
9 <h2>Step 2.1 What is HTML?</h2>
10
11 <p>A Markup Language is a computer language that uses 1
12
13 <p>So, HTML is a markup language specifically meant for
14
15 <p>HTML pages can exist both off- and online. It is ful
16
17 <p><u>Download the Step 2.1 page of this tutorial on yc
18
19 </body></html>
```

You will see something like this in Notepad++ (note that the automated colored formatting may not be present in other text editors).

The document consists of content contained between **tags**. An **opening tag** is a specific bit of code contained between chevrons, such as `<html>` and a **closing tag** with the same code, also between chevrons and also including a forward slash: `</html>`.

The computer, or in particular your browser, knows how to interpret that code and uses this to structure html documents.

An html document has a “**nested**” **structure**, where the `<html>` tag is always the outside of the nest and the different sections are contained within it, which themselves may be the outside container for tag-based elements. For example, in the 2.1 html file the “Download the Step 2.1 page of this tutorial on your local machine and inspect it using Notepad++” text is contained within all of these 4 different tags:

```
<html>
  <body>
    <p>
      <u> Download the Step 2.1 page ... inspect it using
      Notepad++ </u>
    </p>
  </body>
</html>
```

There are a lot of html tags that can be used for many different things. We will play around with some in this tutorial, but for an extended overview, see the [W3 schools html5 tutorial](#). There is also an overview of all current tags to be found at [W3 Schools](#).

A very nice interactive html tag overview can be found at:

<https://html-css-js.com/html/tags/>

Of course, by looking at the tag within the browser, you can also figure out what it is used for.

What is the `<u>` tag used for?

Step 2.3 Manipulating HTML

Of course, you can do more than just reading HTML, you can also use it to write with and thereby change the content and appearance of an html page.

We have opened the 2.1 html with a text editor, but of course you can still open this webpage with your browser. In fact, it is highly likely that this is the default way of opening .html files.

Open 2.1 html with your browser and it will display as it did when it was still online.

Now, get back to the text editor. Let's give a title to this html document.

Enter any title you like between the `<title>` tags.

Open the html document on your browser again. What changed?

HTML is a very simple but quite powerful language, allowing you not only to change the content of an html document, but also its basic looks.

At the same time, it is, like any computer language, **formal** (a computer science word for very strictly bound to a specific definition and format) in its use. If you don't use the tags right, browsers may simply not know how to read your content and structure it incorrectly.

Go all the way to the bottom of your text file.

Type some text, e.g.

followed by a hard return (an Enter), and another line or so, as if you would start a new paragraph e.g.

```
<p style="margin-left:18.0pt;">What you see now is the raw html view of the webpage you downloaded.</p>
</body></html>
Some more stuff here outside of the html tags...
Even more stuff down here below that other stuff!!!
```

Open it in your browser, how did it read your text?

(Hint, if you want to fix this: copy the text within the `<body>` tags as well as use `<p>` to create separate paragraphs out of your text)

Are there easier ways to inspect webpages?

Yes, you will find 'Developer Tools' in Chrome *Chrome button* → *More Tools* → *Developer Tools* (or Shift+Ctrl+I.)

Alternatively, you can right click on any element in a webpage and select *Inspect*.

In the sidebar, you'll get the same view that you'll have when downloading the page and inspecting it in a text editor. Try and do this for even a slightly less barebones webpage than those this tutorial and you'll quickly get an understanding for the complexity of and abundance of html tags in webpages.

Step 2.4 My first HTML.html!

By now you should have a general idea about how HTML works. The best thing to do now, is to create your very own first html page.

Open a new text file, save it as “my first html.html”

Create a HTML text containing:

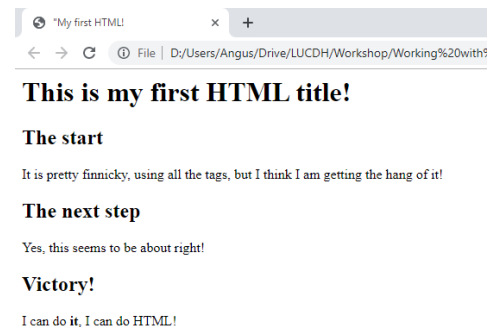
- At the top a HTML doc-type declaration: `<!DOCTYPE html>`
 - Why? Convention: telling the reader (human or computer) what type of document they are dealing with is part of the HTML standard and good practice in general.
- a head of the page `<head>`
 - a title
- a body
 - at least one level 1 header.
 - 3 paragraphs, each with their own level 2 header.
 - Of which at least one has some bold or italic text

Make use of the interactive tag searcher at

<https://html-css-js.com/html/tags/>

or any other HTML cheatsheet, if you need to.

At the end your first HTML should look something like this in your browser.



Step 2.5 HTML elements and attributes

The tags and the content between tags is what is referred to as an **element**. So,

```
<p>I can do it, I can do HTML!</p>
```

contains two elements, one between `<p>` and `</p>` and the other `it`, encapsulated in the `<p>` tags.

HTML, like many other markup languages, also has **attributes**. These provide additional information about html elements. The most prominent example of attributes in HTML documents is also their defining feature, our buddy from part I of this tutorial: the **hyperlink**.

```
<a href="https://workweb.lucdh.nl/tutorial ">Start of the tutorial</a>
```

Add this element and its attribute to your “my first html” file, somewhere at the top (after the Doctype declaration and the title).

If you typed this correctly, you should now be able to return to the start of the tutorial in the browser form of your html document.

Note what happens: while your document is offline, your computer is (or should be) connected to the internet. Your browser knows how to retrieve the page that was linked from an offline document. This does not work the other way around: if you link from an online file to a file on your computer, browsers from around the world can't magically get access to your offline html files. This will result in a **broken link** or a 404 page.

Another very useful and prevalent HTML attribute is ``. This allows you to link to image documents somewhere on the web.

```

```

As you can see, this tag operates with two attributes: `src` and `alt`.

Add this element and its attributes to the end of your document (before `</body>`).

Finally, before starting Step 2.5, save a back-up copy of “my first html” in your same folder.

Step 2.5 Styling HTML documents using CSS

Title, headings, paragraphs, links, and images. This is already starting to look like a real webpage!

Sort of... If we uploaded our “my first html” html document to a server, for all intents and purposes it would be an actual webpage. But this is not the 2nd millennium! We are used to have our webpages with a bit more style.

For that we can quite literally use the `style` attribute.

```
<tagname style="property:value;">
```

A style attribute can be used with an opening tag to change the entire style of that element at once. To do that it uses a specific **CSS property** and a **CSS value**.

CSS stands for Cascading Style Sheets. It quite literally functions as a cascade of stylistic information draped over HTML elements.

If HTML is the walls, the floors, and ceilings of a structure, CSS is the paint on the walls, a nice warm rug on the floors, and the crystal chandelier on the ceiling. This metaphor — CSS running from a basic coat of paint to rich decoration — is meant to underline the enormous variability of style you can achieve with CSS properties and values. Almost everything you see on the web is a combination of HTML and CSS.

Let’s try and decorate ‘my first html’.

Text color can be changed with the property `color`, value can be a color name, such as `blue`. Alternatively you use hexe-colorcodes or rgb (red, green, blue numbers): `#0000FF` or `rgb(0,0,255)` for blue.

Change your heading level I text to blue

Add `style="color:blue;"` in the opening tag.

The result should look something like this:

[This links to the start of the tutorial](#)

This is my first HTML title!

The start

It is pretty finicky, using all the tags, but I think I am getting the hang of it!

Almost everything?

Almost, yes. HTML and CSS is a very powerful combination. However, websites also (more and more) have moving parts or other complex aspects. These sorts of things are made possible with the use of a programming language, in particular [JavaScript](#).

JavaScript is a really (relatively speaking) easy programming language to learn and is quite useful for a range of situations. It is way beyond the scope of this tutorial, but, if you are looking to get into programming, this is a really good and fun language to pick up.

If you master HTML, CSS, JavaScript and a ‘back-end’ language (the language that allows websites to talk to servers, also beyond the scope here), you can call yourself a ‘**full-stack**’ web developer, i.e. you have everything you need to independently create websites from scratch.

What happens if you add `style="color:red;"` to `<body>`?

As you can see, CSS information quite literally cascades, starting from the top, then going down, sequentially taking the next piece of style information, 'painting over' whatever style was applied previously.

Try experimenting with other CSS properties, such as `font-size` (taking a number followed by `px`, for pixels, as value) or `background-color` (also taking a color as value). You can find an overview at: <https://www.w3schools.com/css/>

Hint: you can directly add other style properties and values, following a `;`

If you experiment a bit, you may get something as... cheery as this:



Step 2.6 Referencing external style sheets.

The CSS we have been applying in step 2.5 is referred to as 'inline CSS'. Once again, this quite literally (computers and people working with them take almost everything quite literally) means a style attribute in the lines of the document we are writing.

One webpage may still be feasible, but suppose you have to such inline styling for an entire website? That would take ages, even for a simple website.

That's why it is also possible to make one external style sheet and refer to it in the `<head>` element of the html page:

```
<link rel="stylesheet" href="stylesheet.css">
```

Do so for your "copy of my first html" file.

Remember `href`? This means this attribute will refer to a document following this link. It could therefore be used to refer to a style sheet that is online somewhere. However, since our document is offline, a link that does not connect to a document on the web (so without `http://` in front of it) means it will search for a document in the same folder with this exact name.

So... Why does this example HTML look like hot trash?

There's two simple reasons for this:

- (1) I am not a (web) designer, likely neither are you. Designing a website's look and feel, its **User Interface** or **UI**, is a job all its own: UI-designer. It will take quite long to get to the level of a professional, even if you have a natural aptitude for it (I do not).
- (2) We're just playing around here. Remember, the goal is to understand the basic building blocks of a webpage, not to design an entire webpage. Doing so takes days, or, depending on the complexity of the project and the wishes of the customer, even years.

Let's give it something to link to! Create a "stylesheet.css" file in the same folder as your "copy of my first html" file.

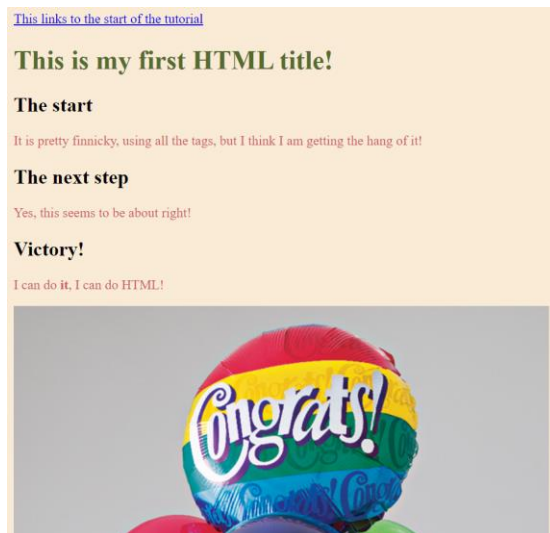
When this file has been created, open it with notepad++ (right click, open with notepad++)

An external style sheet is simply a long list in a specific format, that first identifies specific HTML tags and then provides the CSS property and value that should be applied to it in brackets { }.

For example,

```
body {  
  background-color: AntiqueWhite;  
}  
h1 {  
  color: DarkOliveGreen;  
}  
p {  
  color: IndianRed;  
}
```

Will give you a page that looks like this:



Fill out your own stylesheet.css file with properties and values of your own choosing.

Now, for whatever html file you would create, if you simply link to stylesheet.css, your pages will have this exact same style. Of course, if you would be developing a website, different sections of it may link to different external stylesheets.

If there is time, why not play around some more by adding more tags and/or CSS properties to your stylesheet.css file?

Step 2.7 Congrats!

Congrats! You now know about all the basic building blocks of an HTML document and by extension a website. Still, I think you will also understand that actually using all these building blocks to make a website would be quite a tough and time-consuming challenge.

Fortunately, all of this, and much more, can be handled by a **Content Management System**, or **CMS**. Next up, we will have a look at **Wordpress CMS**

Part 3: Working with Wordpress CMS

Step 3.1 What is Wordpress?

Wordpress is a content management system for websites, although, for once, it is about much more than managing content of websites.

Wordpress started out as a blogging platform in 2001, but it turned out to be such an efficient way of creating and managing your own websites that it rose to become the most popular content management system on the Web: an estimated 34% of all websites run Wordpress. In other words, 1 in every 3 websites you visits runs on Wordpress.

One reason for this is that it is easy, but also incredibly powerful. There are a lot of **plugins** for Wordpress (i.e. ways of extending its basic functionality) and a lot of different, ready to go styles, called **themes**, as well.

But why not see for ourselves?

Go to <https://workweb.lucdh.nl/>

This website was created in less than the time it took to write part I of this tutorial. (It did involve using pre-made work from others, more on that later).

Next, go to <https://workweb.lucdh.nl/login>

Click the register link and register with a username of your choosing and email address you have access to.¹

Check your email and create a new **strong** password for yourself.

Now login to Wordpress

What you will see is part of the **'back-end'** of a website (which also includes the server, the database, and the software that communicates between it). It is the place where the content management happens. What website users see is referred to as the **'front-end.'**

There is not much to this back-end?

Yes, you do not see everything Wordpress CMS has to offer now. On the left side of your screen you will 'only' see posts, events, media, comments, contact, projects, and profiles items.

This has to do with the fact that you were registered as **Authors**, which is a specific **role** in Wordpress that allows you to create some types of content in the back-end, such as **posts** (blog-style webpages).

Admins will have control of over the entirety of the CMS, but, for security reasons, it is not possible to provide you with this role for this tutorial.

Wait... What? Wordpress can do all of the stuff that I just learned for me? And more? Why did I just go through all of that?

Good question! I believe that, while it is cool and easy that so many of the technical aspects of the digital media we create are handled for us, it is also quite important to have a basic understanding what is going on under the hood of all those beautiful websites.

Why?

Well, because being 'digitally literate' will give you an edge in today's fully digital society.

For example: if you understand the amount of work, knowledge, and skill that goes into creating a single webpage, you will also be able to communicate more professionally about the cost in time and/or money that is involved in the development of a website.

In addition, while the more complex stuff will always be difficult or time-consuming for a non-specialist, there may be small repairs or upgrades you could do to websites or webpages with even a basic understanding of HTML and CSS.

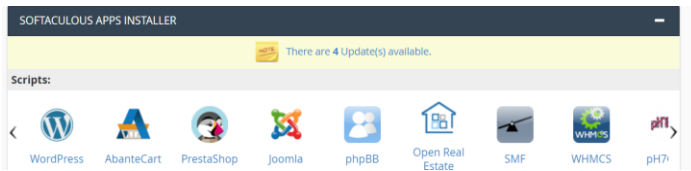
¹ Please note, that by registering you provide the course instructors with the right to process your personal data. Under the **General Data Protection Regulation**, your email and all other data you may enter (e.g. blog posts) for this tutorial will be saved maximally for up to two years or until you revoke the right to process your data.

Step 3.2 Installing Wordpress

We will work a bit with Wordpress posts and projects later, but for now, sit back and enjoy an interactive demo on how to very easily install Wordpress. (The reason you will not be able to work along is that this requires you to have administrative access to a server, which is not possible due to security reasons.)

Remember the **CPanel**? This was the control panel that allows you to manage your server remotely. It allows us to do many things, including uploading individual html documents to a file like structure, as I did for part 2 of this tutorial:

One of the other very easy features available in most CPanels is *Softaculous*, which is an easy install tool for a wide range of frequently used web apps, including Wordpress.

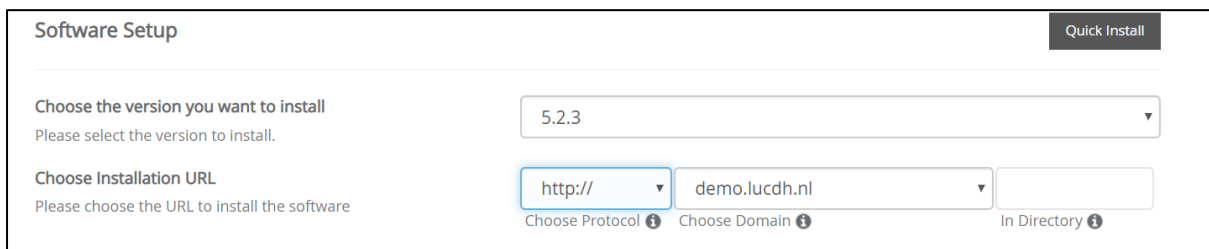


Can you also install Wordpress without the use of Softaculous?

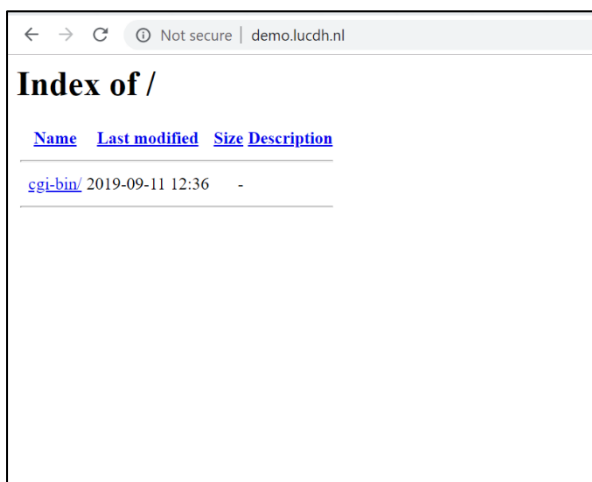
Yes, this is explained at the Wordpress [support site](#).

It involves setting up a database for Wordpress on your web server and the uploading of the Wordpress CMS to that part of your website where you want to have it function. If you're serious about web development, it is definitely worth checking out once, but it is quite time consuming and a bit more complex than we have time for

Since Wordpress is already installed on the `workweb.lucdh.nl` subdomain, for this demo we will install it in `demo.lucdh.nl`.

A screenshot of the 'Software Setup' form. It has a 'Quick Install' button in the top right. The form contains two main sections: 'Choose the version you want to install' with a dropdown menu set to '5.2.3', and 'Choose Installation URL' with three input fields: 'http://', 'demo.lucdh.nl', and an empty 'In Directory' field. Each field has a small information icon.

Before the install:



After the install:

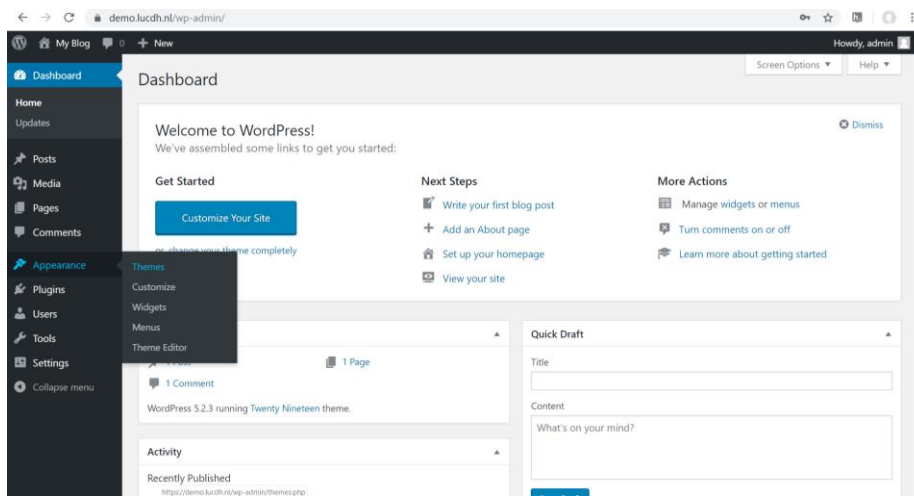


Step 3.3 Acquiring and Setting up Wordpress Themes

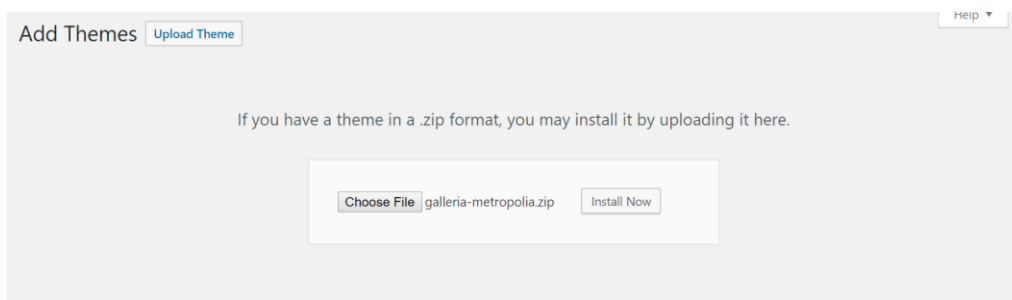
The Wordpress website we just installed is completely functional, but the default **theme** is still very bare bones. It would be completely possible to edit the default theme's Cascading Style Sheets ourselves. However, many people simply choose to acquire a ready-made **theme**, with CSS (and JavaScript features, and plugins) already pre-developed by someone else.

Although it is possible to find very good free themes, generally speaking you need to pay a bit for higher quality or higher functionality themes. One place to do this is at themeforest.net, which is where I acquired the [Galleria Metropolia](#) theme. As the (original) context for this tutorial is a course on Digital Exhibition, I choose it because it mimicked the feel of an online art gallery (it has a lot of extended functionality that we will not use for the context of this course, such as an online art webshop). This is also the theme that is installed at workweb.lucdh.nl, but I will show you how to install this in Wordpress (only possible with Admin role).

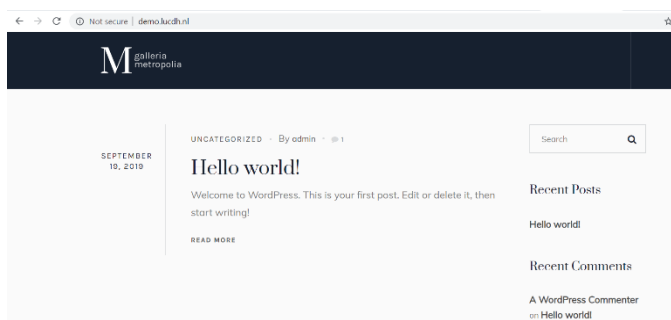
→ Appearance → Themes → Add New



Add Themes → Upload Theme → Choose file → Browse to location of zip file containing the theme. → Install Now → Wait for install → Activate



Result:

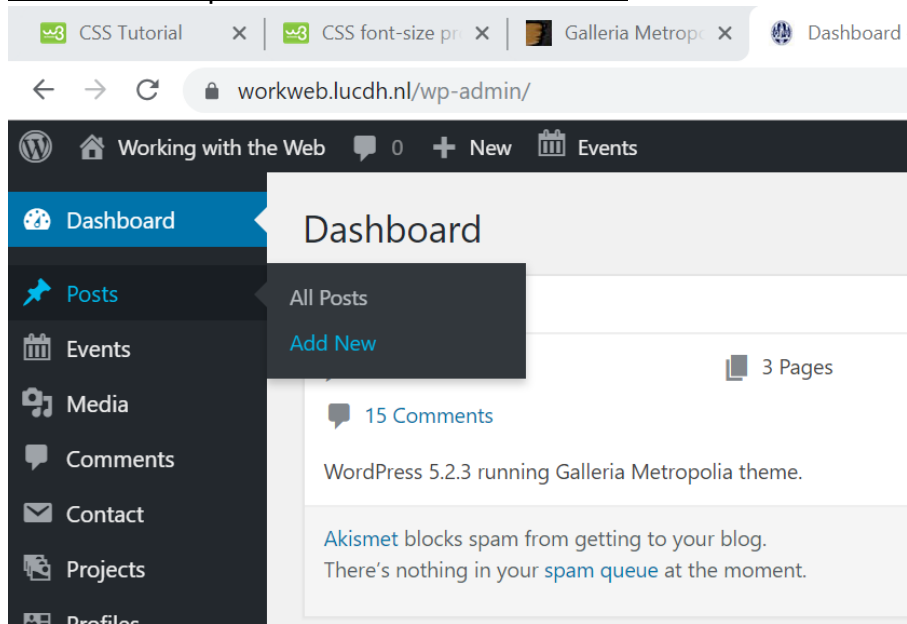


Note that this theme does not automatically look like the website at workweb.lucdh.nl. This requires a bit of tinkering and setting up (for which we don't have time), but all the tools needed for doing so are now installed.

Step 3.4 Creating posts

Let's put you back in the driver's seat!

Go to the Wordpress back-end → Posts → Add new

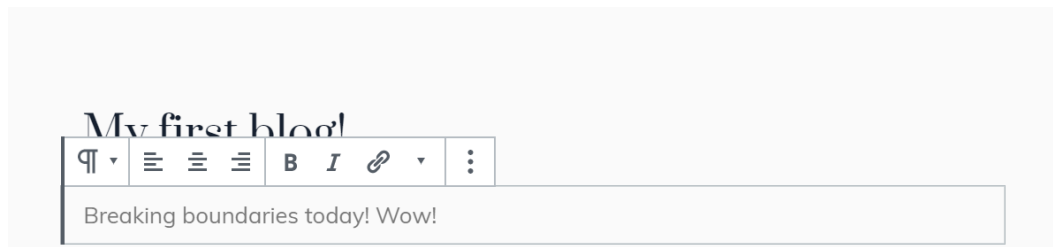


Add a title for your first blog!

Start writing! A sentence at first.

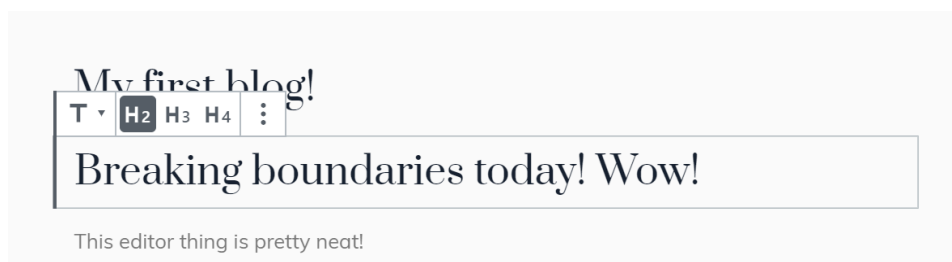
Then hit Enter, and write another sentence

You'll see that, as soon as you start writing, Wordpress recognizes that you are writing a paragraph, indicated by the pilcrow symbol (¶)



Since 2018, Wordpress has a new content editor, called Gutenberg. It is based on adding blocks of content. Often it will try to automatically recognize what sort of content you are trying to add, but of course you can also change this to other types of content you want.

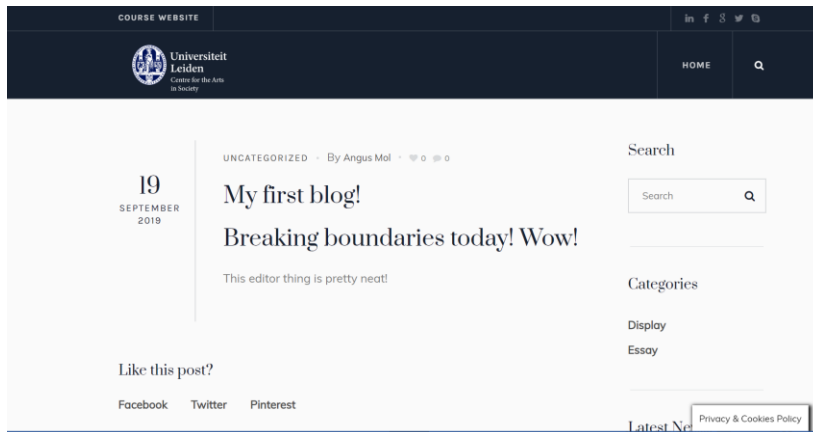
Change your first sentence into a H2 heading (click on it and it should show you the content option menu).



You can save your draft or preview what your post will look like by hitting the preview button in the top right corner.



Preview your blog. It should look something like this.



Step 3.5 HTML Redux!

Remember that HTML thing you learned 30 minutes or so ago? It's still there, but just hidden from our eyes due to the power of Wordpress. We can unhide it, however.

Click on the three vertical dots in the top right corner (next to the *Publish* and the *Gear* button).

Right now the *Editor* is set to *Visual Editor*. Set it to *Code Editor*

```
<!-- wp:heading -->
<h2>Breaking boundaries today! Wow!</h2>
<!-- /wp:heading -->

<!-- wp:paragraph -->
<p>This editor thing is pretty neat!</p>
<!-- /wp:paragraph -->
```

There it is again!

HTML, the universal language of the World Wide Web!

Which you have already mastered the basics of: well done!

What is that <!-- /wp:paragraph --> code?

Everything between <!-- and --> is another HTML tag, known as a Comment. What this special tag does is tell the browser to ignore everything between its opening and end. This allows for the writing of comments that will not get displayed on a webpage. Developers use it all the time to leave messages for themselves and others as to the functionality of certain bits of code.

But these comments are generated by Wordpress?

Good point, my Google Fu failed me when I tried to find out why Wordpress leaves all these seemingly unnecessary comments.

Step 3.6 Onwards and upwards!

You may wonder what to do now?

I suggest you do the following, during the coming week:

Try and get more acquainted with Wordpress CMS. One cool way to do so would be to finish your first blog. You could, for example, write a short blog on your favorite (digital) artist or museum?

You can and should try to use images in your blog. It is just another type of content block (or if you're feeling brave, you could try adding one in using HTML and Wordpress' code editor).

Don't forget to hit *Publish* when you are satisfied.

Do remember that whatever you end up publishing can, in theory, be seen by everyone with access to the WWW.

(no, worries: you can always unpublish, by switching it back to a draft).

Alternatively, under Projects, you could even create a little sample of some of your favorite works from that artist or museum.

If you have any questions or run into any challenges, I'll try to answer them at the start of next week's tutorial!

Can I just use any image I find online?

Another question for the ages!

The answer is: no!

While you are always allowed to link to any content online, uploading an image to your own server and then using it in a web document is, ultimately, an infringement of **copyright**.

Does copyright apply to other content as well?

Yes!

What content is safe to use?

Images or other content that you own copyright over and content over which the original creator has waived copyright or otherwise clearly labelled for re-use.

There are a couple of useful tools out there to help you find those types of media:

- In Google's Image Search under Tools, you will find Usage Rights. Select whichever reuse is appropriate and search for images.
- In [Europeana's](#) Collection Search, you can also use a *Can I Use It?* feature.
- All of the media on [Wikimedia](#) has clear usage rights information.

Finally, consider contributing to the free use of cultural expressions, by sharing images or other content you