HTML

**1)HTML is the standard markup language for documents to be displayed in the browser.**

Versions of HTML

HTML 1.0

The original version of HTML was HTML 1.0. It had very limited features which greatly limited what you could do in designing your web pages.

HTML 2.0

HTML 2.0 then arrived and included all the features of HTML 1.0 plus several new features for web page design. Until January, 1997, HTML 2.0 was the standard in web page design.

HTML 3.0

HTML 2.0 served its purpose very well, but many people designing web pages (called HTML authors or webmasters) wanted more control over their web pages and more ways to mark up their text and enhance the appearance of their websites. Netscape, the leading browser at that time, introduced new tags and attributes called the Netscape Extension Tags. Other browsers tried to duplicate them but Netscape did not fully specify their new tags and so these extension tags did not work in most other browsers. It led to considerable confusion and problems when HTML authors used these tags and attributes and then saw that they didn't work as expected in other browsers.

HTML 3.2

As more browser-specific tags were introduced, it became obvious that a new standard was needed. For this reason, the Word Wide Web Consortium (W3C), founded in 1994 to develop common standards for the evolution of the World Wide Web, drafted the WILBUR standard, which later became known as HTML 3.2. HTML 3.2 captures the recommended practice as of early 1996 and became the official standard in January, 1997. Most, if not all, popular browsers in use today fully support HTML 3.2.

HTML 4.0

In the early days, HTML 4.0 was code-named COUGAR. This version introduces new functionality, most of which comes from the expired HTML 3.0 drafts. This version became a recommendation in December, 1997 and a standard as of April, 1998. Explorer has done a very good job in implementing the many features of HTML 4.0. Unfortunately, Netscape has not kept pace. The latest version of Netscape Communicator still does not recognize the many tags and attributes introduced with HTML 4.0. This means that a web page that involves HTML 4.0 specific tags will look great in Explorer but can look disastrous in Netscape.

XHTML

You would think that the next major version after HTML 4.0 would be HTML 5.0 and with it would come a bunch of new tags that would do all sorts of wonderful things. That would be a good guess - but it would also be a wrong guess. The next version of HTML after HTML 4 is XHTML.

XHTML stands for EXtensible HyperText Markup Language.

XHTML is not bringing with it a lot of new tags. The purpose of XHTML is to address the new browser technologies that are sweeping the world. Today web pages are being viewed in browsers through cell/mobile phones, cars, televisions, plus a host of hand-held wireless devices and communicators. Alternate ways to access the internet are continually being introduced. In many cases, these devices will not have the computing power of a desktop or notebook computer and so will not be able to accommodate poor or sloppy coding practices. XHTML is designed to address these technologies. XHTML also begins to address the need for those with disabilities (such as the blind and visually impaired) to access the internet. Thus web pages written in XHTML will allow them to be viewed on a wide range of browsers and internet platforms.

XHTML is the result of the hard working World Wide Web Consortium (the W3C) to bring some sort of standard to provide rich high quality web pages through these varied devices. XHTML became an official W3C recommendation in January, 2000. XHTML is now a web standard and is the next generation of HTML.

HTML 5

HTML 5 (usually written HTML5) is the new web standard. It follows HTML 4 (which came out way back in 1997) and XHTML. Since the introduction of HTML4, a lot has happened with the web and something needed to be done to address all the new technologies and latest multimedia. HTML5 is the result of cooperation that began in 2006 between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). While HTML5 is still evolving (still under development), the latest browsers do support many of the new features and elements in this version. The basic aim of HTML5 is to provide two things – (1) to improve the language and (2) to support the latest multimedia. In order to accomplish this, some ground rules were established by the W3C and WHATWG. Among them were to reduce the need for external plug-ins (such as Flash plug-ins), better handling of errors, and more markup elements (tags) to replace scripting. HTML5 should also be device independent (that is, understood by computers and the many devices in existence today) while also keeping it easily readable by us humans.

* SVG, canvas and other virtual vector graphics are supported in HTML5, whereas in HTML, using vector graphics was only possible by using it in conjunction with different technologies like Flash, VML, and Silver-light, etc.
* HTML5 uses web SQL databases, application cache for temporary storing data, meanwhile, in HTML, only browser cache could be utilized for this purpose.
* Another difference between HTML and HTML5 worth mentioning is that the former doesn’t allow JavaScript to run within the web browser (it instead runs in the browser interface thread) whereas the latter provides full support for JavaScript to run in the background (This is possible courtesy to the JS web worker API of HTML5).
* HTML5 is not based on SGML, and that allows it to have improved parsing rules which provide enhanced compatibility.
* In HTML5, inline MathML and SVG can be used in text whereas this wasn’t possible in HTML.
* Some of the deprecated elements that have now been dropped completely are: isindex, noframes, acronym, applet, basefont, dir, font, frame, frameset, big, center, strike, tt.
* HTML5 supports new kinds of form controls, for example: dates and times, email, number, range, tel, url, search etc.
* There are many new elements introduced in HTML. Some of the most important ones are: summary, time, aside, audio, command, data, datalist, details, embed, wbr, figcaption, figure, footer, header, article, hgroup, bdi, canvas, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, track, video.

**Key Advantages Provided by HTML5 for Developers**

HTML5 wanted developers to have more flexibility while designing websites and there have been significant improvements that are worth noticing:

Persistent error handling:

Most of the browsers have the support to parse structurally/syntactically incorrect HTML code, but until a few years ago, there was no standardized process to handle this. It meant that new browser developers had to perform malformed HTML document tests in different browsers in the bid to create improved error handling processes via the marvels of reverse engineering. The consistent HTML5 error handling has made a huge difference in this regard. The improved parsing algorithms that are used in HTML5 have unquantifiable benefits. Surveys reveal that around 90 percent of websites are liable to comprise of some incorrectly written code (tag soup) so being able to handle errors aptly is imperative. Furthermore, inherent error handling saves developers a lot of money and tons of time.

Improved semantics for elements:

To enhance code insinuation, improvements have been made to the semantic roles of various existing elements. Section, article, nav and header are the new elements that have replaced most of the now-obsolete div elements, and this has made the process of mistake-scanning a whole lot less complicated.

Enhanced support for web application features:

One of the primary goals of HTML5 was to allow browsers to function as application platforms. Web sites in the past used to be a lot less complex but over time, the cumbersomeness has increased. HTML5 provides developers with enhanced control of their websites’ performance. In the past, the developers had to use workarounds because many server-side technologies and browser extensions were not present. Now, with HTML5, there is no use to employ any JS-based or Flash work-around (as previously done in HTML4) because there are elements inherently present in HTML5 that provide all the functionalities.

Mobile web made easier:

Even today, creating a mobile version of a website can be a headache for developers. The smartphone-owning demographic has seen exponential proliferation over the past decade, and that created a need for improved HTML standards. End users want to be able to access a web resource at any time and via any device which makes having responsive websites a requirement. HTML5 has made mobile support a lot simpler by being able to cater to the low-powered electronic devices like tablets and smartphones.

The canvas element:

One of the most discussed features of HTML5 is the <canvas> element. The introduction of this unique tag has had an enormous impact on the use of Adobe Flash. Even though many websites still use Flash, HTML5 has got the preference of many people, and it’s believed that soon Flash will run completely obsolete.Using the canvas element, a developer can draw graphics using different colors and shapes by making use of scripts (e.g. JavaScript). It’s worth mentioning that canvas is merely a graphic container and to define the graphics, a script has to be executed. An example where JavaScript is used in conjunction with canvas is:

<canvas id=”TestCanvas” width=”200″ height=”100″></canvas>

var c = document.getElementById(“TestCanvas”);

var context = c.getContext(“2d”);

context.fillStyle = “#FF0000”;

context.fillRect(0,0,140,75);

The Menu element:

The newly added <menu> and <menuitem> elements are constituents of the interactive elements specifications but aren’t often used or talked about by the developer community. However, these two items can be used to ensure enhanced web interactivity.The <menu> tag is used to represent menu commands in mobile and desktop applications for simplicity purposes. A possible usage of the menu tag can be:

<body contextmenu=”new-menu”>

<menu id=” new-menu” type=”context”>

<menuitem>Hello!</menuitem>

</menu>

</body>

Customizable Data Attributes:

It was possible to add custom attributes before HTML5, but it was a risky affair; for example, in HTML4, custom attributes would sometimes stop a page from rendering completely, and they could often become the cause of incorrect/invalid documents. The data-\* attribute in HTML5 has brought an end to this often-occurring problem. There are multiple uses for this attribute, but the primary objective of its introduction was the storage of extra information about different elements. Now, custom data can be included, and it gives developers increased chances of making engaging and efficient web pages without having to introduce server-side lookups or Ajax calls.

Cookies were (or should be):

Local storage support was a huge addition to HTML5. In the pre-HTML5 days, if developers wanted to store anything, they had to make use of cookies. Cookies can hold a small amount of data (not to mention, everybody hates them) and this made the localStorage object’s addition to HTML5 an even more welcome benefit. The localStorage object is a part of the global window namespace and can be accessed from wherever desired while using scripts.

**2)What happens when you click a website?**

Web pages are written in HTML, Hypertext Markup Language. ... The Web browser renders the page according to the HTML code. Web servers are computers whose job is to respond to a browser's request for a web page and deliver it through the internet. Pages hosted on a web server can be displayed to anyone all over the world.

And when the web link is typed or clicked your browser takes that URL, breaks out the name of the web site, and then uses the Domain Name System (DNS) to get an Internet Protocol (IP) address for the site. Your browser then opens a Transmission Control Protocol (TCP) connection to the web site over IP.

DNS(Domain Name System) is a database that maintains the name of the website (URL) and the particular IP address it links to. Every single URL on the internet has a unique IP address assigned to it. The IP address belongs to the computer which hosts the server of the website we are requesting to access.

1. You type maps.google.com into the address bar of your browser.

2. The browser checks the cache for a DNS record to find the corresponding IP address of maps.google.com.

DNS(Domain Name System) is a database that maintains the name of the website (URL) and the particular IP address it links to. Every single URL on the internet has a unique IP address assigned to it. The IP address belongs to the computer which hosts the server of the website we are requesting to access. For an example, www.google.com has an IP address of 209.85.227.104. So if you’d like you can reach www.google.com by typing http://209.85.227.104 on your browser. DNS is a list of URLs and their IP addresses just like how a phone book is a list of names and their corresponding phone numbers.

The main purpose of DNS is human-friendly navigation. You can easily access a website by typing the correct IP address for it on your browser but imagine having to remember different sets of numbers for all the websites we regularly access? Therefore, it is easier to remember the name of the website using an URL and let DNS do the work for us with mapping it to the correct IP.

In order to find the DNS record, the browser checks four caches.

● First, it checks the browser cache. The browser maintains a repository of DNS records for a fixed duration for websites you have previously visited. So, it is the first place to run a DNS query.

● Second, the browser checks the OS cache. If it is not found in the browser cache, the browser would make a system call (i.e. gethostname on Windows) to your underlying computer OS to fetch the record since the OS also maintains a cache of DNS records.

● Third, it checks the router cache. If it’s not found on your computer, the browser would communicate with the router that maintains its’ own cache of DNS records.

● Fourth, it checks the ISP cache. If all steps fail, the browser would move on to the ISP. Your ISP maintains its’ own DNS server which includes a cache of DNS records which the browser would check with the last hope of finding your requested URL.

You may wonder why there are so many caches maintained at so many levels. Although our information being cached somewhere doesn’t make us feel very comfortable when it comes to privacy, caches are important for regulating network traffic and improving data transfer times.

3. If the requested URL is not in the cache, ISP’s DNS server initiates a DNS query to find the IP address of the server that hosts maps.google.com.

As mentioned earlier, in order for my computer to connect with the server that hosts maps.google.com, I need the IP address of maps.google.com. The purpose of a DNS query is to search multiple DNS servers on the internet until it finds the correct IP address for the website. This type of search is called a recursive search since the search will continue repeatedly from DNS server to DNS server until it either finds the IP address we need or returns an error response saying it was unable to find it.

In this situation, we would call the ISP’s DNS server a DNS recursor whose responsibility is to find the proper IP address of the intended domain name by asking other DNS servers on the internet for an answer. The other DNS servers are called name servers since they perform a DNS search based on the domain architecture of the website domain name.

Without further confusing you, I’d like to use the following diagram to explain the domain architecture.

Many website URLs we encounter today contain a third-level domain, a second-level domain, and a top-level domain. Each of these levels contains their own name server which is queried during the DNS lookup process.

For maps.google.com, first, the DNS recursor will contact the root name server. The root name server will redirect it to .com domain name server. .com name server will redirect it to google.com name server. google.com name server will find the matching IP address for maps.google.com in its’ DNS records and return it to your DNS recursor which will send it back to your browser.

These requests are sent using small data packets which contain information such as the content of the request and the IP address it is destined for (IP address of the DNS recursor). These packets travel through multiple networking equipment between the client and the server before it reaches the correct DNS server. This equipment use routing tables to figure out which way is the fastest possible way for the packet to reach its’ destination. If these packets get lost you’ll get a request failed error. Otherwise, they will reach the correct DNS server, grab the correct IP address, and come back to your browser.

4. Browser initiates a TCP connection with the server.

Once the browser receives the correct IP address it will build a connection with the server that matches IP address to transfer information. Browsers use internet protocols to build such connections. There are a number of different internet protocols which can be used but TCP is the most common protocol used for any type of HTTP request.

In order to transfer data packets between your computer(client) and the server, it is important to have a TCP connection established. This connection is established using a process called the TCP/IP three-way handshake. This is a three step process where the client and the server exchange SYN(synchronize) and ACK(acknowledge) messages to establish a connection.

1. Client machine sends a SYN packet to the server over the internet asking if it is open for new connections.

2. If the server has open ports that can accept and initiate new connections, it’ll respond with an ACKnowledgment of the SYN packet using a SYN/ACK packet.

3. The client will receive the SYN/ACK packet from the server and will acknowledge it by sending an ACK packet.

Then a TCP connection is established for data transmission!

5. The browser sends an HTTP request to the web server.

Once the TCP connection is established, it is time to start transferring data! The browser will send a GET request asking for maps.google.com web page. If you’re entering credentials or submitting a form this could be a POST request. This request will also contain additional information such as browser identification (User-Agent header), types of requests that it will accept (Accept header), and connection headers asking it to keep the TCP connection alive for additional requests. It will also pass information taken from cookies the browser has in store for this domain.

Sample GET request (Headers are highlighted):

6. The server handles the request and sends back a response.

The server contains a web server (i.e Apache, IIS) which receives the request from the browser and passes it to a request handler to read and generate a response. The request handler is a program (written in ASP.NET, PHP, Ruby, etc.) that reads the request, its’ headers, and cookies to check what is being requested and also update the information on the server if needed. Then it will assemble a response in a particular format (JSON, XML, HTML).

7. The server sends out an HTTP response.

The server response contains the web page you requested as well as the status code, compression type (Content-Encoding), how to cache the page (Cache-Control), any cookies to set, privacy information, etc.

Example HTTP server response:

If you look at the above response the first line shows a status code. This is quite important as it tells us the status of the response. There are five types of statuses detailed using a numerical code.

● 1xx indicates an informational message only

● 2xx indicates success of some kind

● 3xx redirects the client to another URL

● 4xx indicates an error on the client’s part

● 5xx indicates an error on the server’s part

So, if you encountered an error you can take a look at the HTTP response to check what type of status code you have received.

8. The browser displays the HTML content (for HTML responses which is the most common).

The browser displays the HTML content in phases. First, it will render the bare bone HTML skeleton. Then it will check the HTML tags and sends out GET requests for additional elements on the web page, such as images, CSS stylesheets, JavaScript files etc. These static files are cached by the browser so it doesn’t have to fetch them again the next time you visit the page. At the end, you’ll see maps.google.com appearing on your browser.

**3)Semantics**

In programming, Semantics refers to the *meaning* of a piece of code — for example "what effect does running that line of JavaScript have?", or "what purpose or role does that HTML element have" (rather than "what does it look like?".)

Semantics in JavaScript

In JavaScript, consider a function that takes a string parameter, and returns an <li> element with that string as its textContent. Would you need to look at the code to understand what the function did if it was called build('Peach'), or createLiWithContent('Peach')?

Semantics in CSS

In CSS, consider styling a list with li elements representing different types of fruits. Would you know what part of the DOM is being selected with div > ul > li, or .fruits\_\_item?

Semantics in HTML

In HTML, for example, the <h1> element is a semantic element, which gives the text it wraps around the role (or meaning) of "a top level heading on your page."

<h1>This is a top level heading</h1>

By default, most browser's user agent stylesheet will style an <h1> with a large font size to make it look like a heading (although you could style it to look like anything you wanted).

On the other hand, you could make any element look like a top level heading. Consider the following:

<span style="font-size: 32px; margin: 21px 0;">Is this a top level heading?</span>

This will render it to look like a top level heading, but it has no semantic value, so it will not get any extra benefits as described above. It is therefore a good idea to use the right HTML element for the right job.

HTML should be coded to represent the data that will be populated and not based on its default presentation styling. Presentation (how it should look), is the sole responsibility of CSS.

Some of the benefits from writing semantic markup are as follows:

Search engines will consider its contents as important keywords to influence the page's search rankings (see SEO)

Screen readers can use it as a signpost to help visually impaired users navigate a page

Finding blocks of meaningful code is significantly easier than searching though endless divs with or without semantic or namespaced classes

Suggests to the developer the type of data that will be populated

Semantic naming mirrors proper custom element/component naming

**4) Grids**

A grid is a collection of horizontal and vertical lines creating a pattern against which we can line up our design elements. They help us to create designs where elements don’t jump around or change width as we move from page to page, providing greater consistency on our websites.A grid will typically have columns, rows, and then gaps between each row and column — commonly referred to as gutters.

Having decided on the grid that your design needs, you can use CSS Grid Layout to create that grid in CSS and place items onto it. We will look at the basic features of Grid Layout first and then explore how to create a simple grid system for your project.

As a starting point, download and open the starting point file in your text editor and browser (you can also see it live here). You will see an example with a container, which has some child items. By default these display in normal flow so the boxes display one below the other. We will be working with this file for the first part of this lesson, making changes to see how grid behaves.

To define a grid we use the grid value of the display property. As with Flexbox, this switches on Grid Layout, and all of the direct children of the container become grid items. Add this to the CSS inside your file:

.container {

display: grid;}

Unlike flexbox, the items will not immediately look any different. Declaring display: grid gives you a one column grid, so your items will continue to display one below the other as they do in normal flow.

To see something that looks more grid-like, we will need to add some columns to the grid. Let's add three 200-pixel columns here. You can use any length unit, or percentages to create these column tracks.

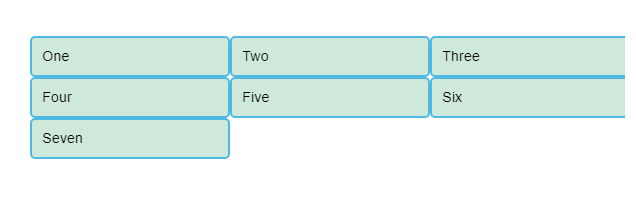
.container {

display: grid;

grid-template-columns: 200px 200px 200px;

}

Add the 2nd declaration to your CSS rule, then reloads the page, and you should see that the items have rearranged themselves one into each cell of the created grid.



**Different ways in which you can make a grid in HTML.**

The implicit and explicit grid

We have only specified column tracks so far, and yet rows are being created to hold our content. This is an example of the explicit versus the implicit grid. The explicit grid is the one that you create using grid-template-columns or grid-template-rows. The implicit grid is created when content is placed outside of that grid — such as into our rows. The explicit and implicit grids are analogous to the main and cross flexbox axes.

By default, tracks created in the implicit grid are auto sized, which in general means that they are large enough to fit their content. If you wish to give implicit grid tracks a size you can use the grid-auto-rows and grid-auto-columns properties. If you add grid-auto-rows with a value of 100px to your CSS, you will see that those created rows are now 100 pixels tall.

.container {

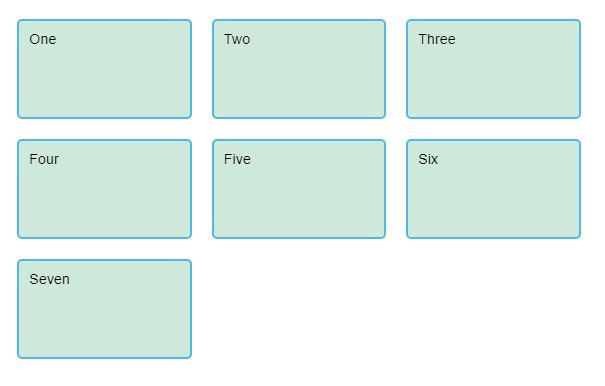
display: grid;

grid-template-columns: repeat(3, 1fr);

grid-auto-rows: 100px;

grid-gap: 20px;

}



Line-based placement

We now move on from creating a grid, to placing things on the grid. Our grid always has lines, these lines start at 1 and relate to the Writing Mode of the document. We can place things according to these lines by specifying the start and end line. We do this using the following properties:

* grid-column-start
* grid-column-end
* grid-row-start
* grid-row-end

These properties can all have a line number as the value. You can also use the shorthand properties:

* grid-column
* grid-row

These let you specify the start and end lines at once, separated by a / — a forward slash character.