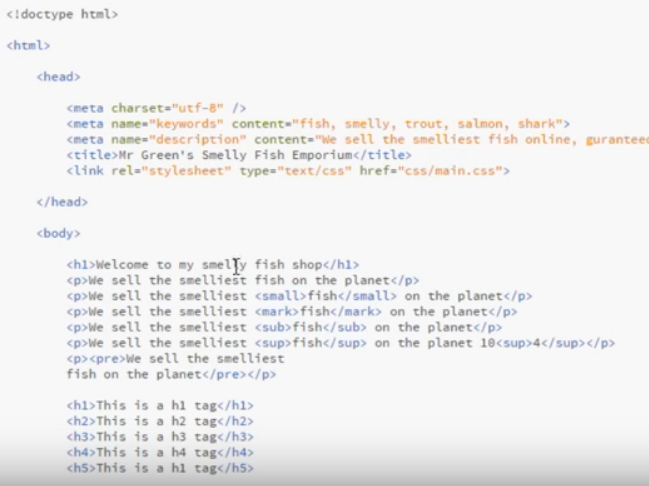
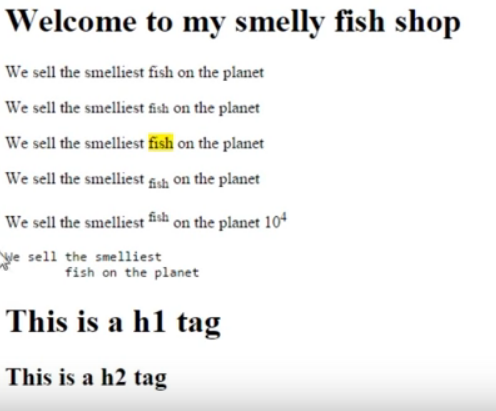
HTML: Stands for Hyper Text Markup Language. It is not a programming language but a markup language; a way of presenting content in a structured, logical way to a user.  
So HTML gives the structure of the web page, CSS (stands for cascading style sheets) is responsible for the styling of the web page and JavaScript makes the web page interactive.

We control the structure of HTML with tags. We use <> tags to mark content up. Tags can contain attributes – they’re giving tags more information. Tags can be nested in other tags. Tags are normally indented when nested for readability.

The <head> tag contains all the extra information about a web page: meta data - key words, character set, content deycription, title, CSS & Java Script references.   
The <body> contains all visual content: Headings, images, paragraphs etc.  
href stands for hyper reference, refers to where the css file is stored.  
Anything you wanna show to the user, goes into the body tag.

<small></small> turns the embedded text small, <mark></mark> highlights it, <sub></sub> stands for subscript, that’s gonna make the text smaller, and push it a little bit down below the text line. <sup></sup> stands for superscript, the shrinks the text and raises it above the text line; <pre></pre> stands for preformatted text – it avoids having the text browser styles.





The <img> tag: wespecify where the image is stored using src attribute. <img src=”image-source-path.jpg”>  
We can specify the width and height of an image. width=”100%” instructs the picture to be the size of the browser window. Usually we specify the width in CSS though.  
alt attributes are usefuls for screen readers or a crawler.

HTML links  
The anchor tag:

* <a href=”link-source”>
* internal – links to pages in own website
* external – links to other websites
* downloads – links to files
* anchors – links to specific ares on a page

Internal links:  
<a href=”contact.html”>Contact Us</a>  
<a href=”prices/freshwater-fisk-prices.html”>Price List</a>  
<a href=”../index.html”>back to index</a> The way we go one step up from a folder and jump out of it (../)

External links:  
<a href=http://www.thenetninja.co.uk>Try out some awesome web tutorials</a> Watch out for the http:// protocol, you need to put it in to specify you’re using this protocol to request this webpage.

Download link:  
<a href=”download/all-prices.pdf”>Download our full price list</a>

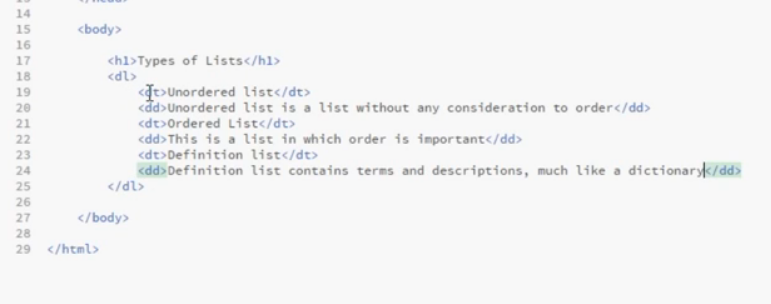
Anchor link:  
<a href=”#top”>to top</a>  
And on the top of the webpage: <div id=”top”></div>

<a href=http://www.thenetninja.co.uk target=”\_blank”>Try out some awesome web tutorials</a> If you want the link to open in a new window, use a target attribute with „\_blank” value

HTML Lists:

* Unordered list: <ul> + <li>
* Ordered list: <ol> + <li>
* Definition list: <dl> + <dt> + <dd> -> we use this one for something, like a dictionary of terms (definition list, definition term and definition description)

A lot of websites use ul for navigation, as a grouping for links for instance.



The div tag:

* stands for „division”
* used to divide content into logical sections

ID’s and Classes:

* An ID is a unique identifier, used once on a page (e.g. header, about)
* A Class is a identifier of an element which can be used multiple times on a page (e.g. comment – when there are multiple comments)
* Both are used for JS and CSS

3 Ways to Add CSS to HTML:

* Inline styling using the style attribute
* Within <style> tags in the head of the document
* By linking up a stylesheet in the head

Linking: <link rel=”stylesheet” type=”text/css” href=”css/main.css”>

CSS:

Classes and ID’s:  
Both are a way to describe your content.

* Classes can be used multiple times on a page
* ID’s can be used only once per page – they are unique!
* you target html classes with a full stop,
* ID’s with a hash sign (#) in front of the selector.

CSS Conflicts & the Cascade:

* The cascade rules from top to bottom
* if the selectors are identical the bottom most rule always wins
* the more specific selector wins

Inheritance:

* every child, grandchild etc. element inherits the parent element’s styling by default
* you can override inheritance with specifying different styling for a child element
* you can override the borwsers default styling

Selector Specificity:

Between  
#main-content p {color: black;}  
and  
p {color: black;}  
the former wins.

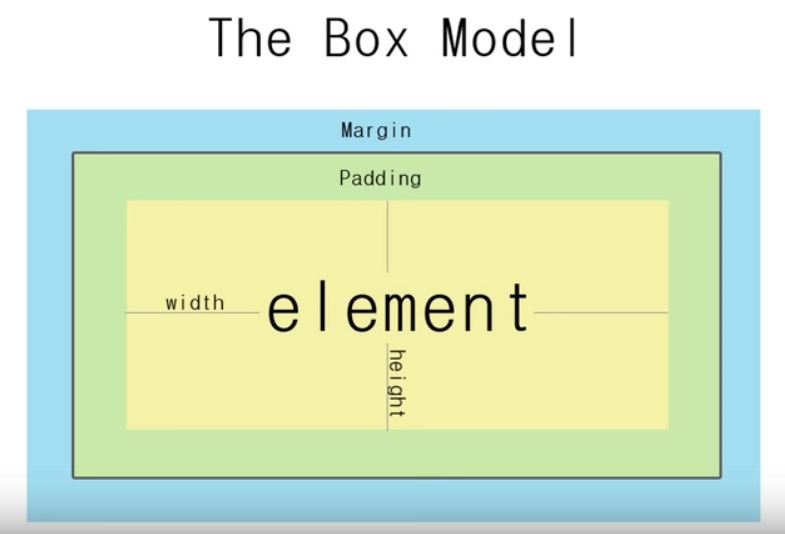
There is a point system for specificity:

ID’s: 100 points Classes: 10 points Elements: 1 point

Targeting Multiple Elements:  
you can separate selectors with commas to apply the same styling to the contributing HTML tags.

Descendant selectors:  
Each tag is a descendant of another another tag if they are nested inside the latter, no matter the level of the nesting.   
If we want to use descendant selectors, first we write down the parent element, then that descendant element of the parent which we want to target.

The Box Model:

It’s a way that elements represent themselves on a page, in terms of space.  


It dictates the spacing of all our HTML elements on the page, as well as element borders. The box model is made up of the element’s width, height, padding ad border.  
*Padding* is an internal spacing, it’s between az element’s edge and the element itself.  
The image itself can have a *Border.*The *Margin* controls the space between the different elements on a page.  
We can also set the *Width* and the *Hight* of the element.

Vertical margins don’t add up, but collapse, and the margin of greater value wins.

If we don’t specify the width of a block level element, it will strech to fill up the width of the browser (apart from the margin, if we specify it).

The code:  
.box {  
 margin: 30px;  
 padding: 30px;  
 border: 1px solid black;  
 width: 200px;  
}

mean an element of 322px wide.

Margins:  
We can specify margins for the different sides of the element.  
.box {  
 margin: 30px 20px 15px 5px;  
}  
The above code specifies margins for the different sides in order of top, right, bottom and left.

If you use  
.box {  
 margin: 30px 20px 15px;  
}  
you specified margins for top, right and bottom, and the left will automatically chose to match the right side’s 20px!  
Therefore margin: 30px 15px; will mean 30px top and bottom margins, and 15px left and right margins.

When you have two block level elements stacked on top of each other, each associated with a margin – one with a margin on the bottom, one with a margin on the top, those two margins will collapse when they meet each other. If they’re identical, they’ll become one margin, and if they aren’t identical, they will take the larger of the two margins.

Margin: auto; is going to position the element in the middle of the browser-width.  
Percentages:  
.box {  
 margin: 30px 25px;  
 border: 1px solid black;  
 width: 50%;  
}  
will have the same effect, in the sense, that it is going to position the element in the middle of the browser window’s width.

Padding:  
Internal spacing property – the space between the element and the border. Specifications work the same way as with margins.  
.box {  
 margin: 30px;  
 padding: 30x 20%;  
 border: 1px solid black;  
}  
You can mix pixels and percentages. Paddings don’t collapse.

Borders:  
The perimeter of the element, just outside the padding, but before the margin. We can specify three values for it, the width, the style and the color.  
.box {  
 margin: 20px;  
 padding: 20x;  
 float: left;  
 border-top: 10px solid blue;   
}

HTML Heading Structure:

* Heading tags range from h1 to h6
* We give sections of a page an appropriate heading depending on its hierarchy in the document

Line breaks and horizontal rules:  
Line breaks are one way to control whitespace on a a webpage.  
<br> inserts a new line in the text. It is self-closing.  
<hr> stands for „horizontal rule”. It is self closing as well. Simply inserts a horizontal line.

Adding JavaScript to HTML:  
The default script language of websites is JavaScript.

* In the <script> tag in the head or body
* By linking a .js file in the head or body (yessss, girl)
* Inline (not a good practice)

First method:  
<head>  
 <script>  
 alert(„Yo Ninja, welcome to my website!”)  
 </script>  
</head>

If you put it at the end of the body part, the browser will first render the full page, and then the pop-up window.

Second method:  
<head>  
 <script src=”script/main.js”></script>  
</head>  
This time the pop-up window will appear first, since we linked our js file in the head. Then it will load the rest of the page. Best practice is to link your script in the bottom of the body tag.

CSS Inline & Embedded Styles:  
Ways to Add CSS:

* Inline styling – Hell No
* Embedded Style Sheets – NOPE
* External Style Sheets – You Go Bitch

Inline styling:

* Time consuming to style multiple elements
* Tricky to manage and update
* Messy
* Good for very specialised rules

Embedded Style Sheets:

* Time consuming to apply global styles
* Hard to maintain and update
* Good for making specific page styles

External Style Sheets:

<head>  
 <link rel=”stylesheet” type=”text/css” href=”css/syntax.css”>  
</head>

CSS Comments and Where to Use Them:  
The syntax of the comment: /\*this is a comment\*/  
Comments for yourself, to other developers, reminders.  
To structure, organize the code.

CSS the Important Declaration:  
The important declaration overrides every and each other declaration.  
Syntax:  
p {  
 color: read !important;  
}  
Try not to use this declaration.

CSS Child Selectors:  
#main-content > p {  
 color: red;  
}  
We select only direct children (e.g. doesn’t affect grand-children.)

nth Child Pseudo Classes:

<article> úgy működik mint egy speciális <div>. összefog egy összetartozó tartalmat.  
<aside> olyan tartalom, ami nem szorosan kapcsolódik az éppen megjelenő oldal tartalmához. Például az oldal szélén ajánlott cikkek.  
<header> fejléc  
<main> represents the main content. kb mint egy második <body>.  
<footer> lábjegyzet  
<span> inline, kisebb egységekre jobb, mint a div  
<section> nagyobb blokkot fog össze mint az article  
Szóval kb main > section > article > div

dl – dd – dt : szemantikai szerepe van a listáknál. Kérdés – felelt szerkezet strukturálására jó kb.

Vannak fun Specificity cheat sheetek.

<blockquote>  
The HTML <blockquote> Element (or HTML Block Quotation Element) indicates that the enclosed text is an extended quotation. Usually, this is rendered visually by indentation (see Notes for how to change it). A URL for the source of the quotation may be given using the cite attribute, while a text representation of the source can be given using the <cite> element.

<q>  
The HTML <q> element indicates that the enclosed text is a short inline quotation. This element is intended for short quotations that don't require paragraph breaks; for long quotations use the <blockquote> element.

<hgroup>  
The HTML <hgroup> element represents a multi-level heading for a section of a document. It groups a set of <h1>–<h6> elements.

<nav>  
The HTML <nav> element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.

REVIEWS:

DOCTYPE-ot kicsivel kell írni?!  
:before, :after-rel lehet üres tageket generálni  
definition list headerek (h3, h4) helyett az anakines cvben?

demo: 4 perc fejenként, mit találtunk a legfonosabbnak a héten, beszámolni mi vot a legjobb, legizgibb, legnehezebb: 3-3 dolog.   
slides.com vagy prezi google slides  
max 10 szó/slide, max 8 sornyi kódkivágás  
5-8 slide összesen kb jó  
angolul!

Fon Size:  
Absolute(px) or relative(em’s or percentages). E.g. font-size: 3em; takes the parent element’s font size, and multiplies it by three.  
font-size: 50%; would take half of the parent element’s font size.

Font Family:  
The style of the font. Font family stack: we can specify styles, so if the first font-family is not installed in the user’s browser, it can fall back on the second specification of the font.  
font-family: arial, helvetica, sans-serif;

Text decoration:  
a {  
 text-decoration: none;  
}  
removes underline from the link.  
You can only have one text decoration!

Font Weight:  
not all fonts have a bold or lighter version!   
font-family: „Yu Gothic”. If the name consists of two words, you have to use quotation marks.  
You can use key words or numbers to set the font weight: 100-300 is lighter, 400-600 is in the normal range and 700-900 is in the bolder range.

Text Transform:  
Change the letter casing. Capitalize: every word will begin with a capital letter. Uppercase will make every latter of every targeted word upper case. Lower case does the opposite. None removes every text transform.

Text Color:  
text/foreground color // background-color.   
*color* means the foreground (pretty much every time the text) color. You can use key words, hexadecimal or rgb to specify colors.  
background-color targets the background’s color... or just type background

Styling Links:  
a {  
 color: crimson;  
 text-decoration: none;  
 font-weight: bold;  
}

a:hover {  
 color: darkmagenta;  
 text-decoration: underline;  
 background-color: aquamarine;  
}

Letter Spacing and Line Height:  
On block level elements, the line-height property specifies the minimum height of line boxes within the element.  
On non-replaced inline elements, line-height specifies the height that is used to calculate line box height.  
Letter spacing is the spaces between letters. Word spacing is straightfroward.  
Line height is the space between text lines.  
p {  
 font-size: 12px;  
 letter-spacing: 10px;  
 word-spacing: 2px;  
 line-height: 24px;  
}  
Line height is the vertical height of the line. We are not specifying the gap between the lines but the height of the whole line.  
Em’s:   
p {  
 font-size: 12px;  
 letter-spacing: 0.2em;  
}  
The em’s here would compare to the font-size’s 12px, instead of the paren’t element’s font size. Same shit for word spacing and line-height, when using em’s.

Pragraph Spacing:  
We’re using margins here.  
p {  
 font-size: 14px;  
 line-height: 2em;  
 margin-bottom: 32px;  
}

Block-level elements:  
You can use the box model on block level elements( which has five properties: margin, border, padding, width and height).  
.block {  
 padding: 10px;  
 margin: 10px;  
 border: 1px solid #000;  
}

.inline {  
 padding: 10px;  
 margin: 10px;  
 border: 1px solid #000;  
}

The block class belongs to divs and the inline class to span elements.  


Block level elements take up a whole row in a document.  
Inline elements stick next to each other in one line. they can only be controlled with the box model horizontally! WE can avoid it by changing its display type:  
.inline {  
 padding: 10px;  
 margin: 10px;  
 border: 1px solid #000;  
 display: block;  
}  
This way they won’t stak from left to right but on top of each other.  
Say we want to keep the block and the inline properties too:  
.inline {  
 padding: 10px;  
 margin: 10px;  
 border: 1px solid #000;  
 display: inline-block;  
}  


a {  
 padding: 10px;  
 margin: 10px;  
 background-color: #ccc;  
 border: 1px solid #aaa;  
 color: #000;  
 text-decoration: none;  
}  
an a tag is an inline element.

Width and Height:  
<div class=”static-width”>Static Pixel Width</div>  
<div class=”percentage-width’>Percentage Width</div>  
<div class=”inline-block”>inline block</div>  
<div class=”inline-block”>inline block</div>

...

#main-content div {  
 background-color: #ccc;  
 margin: 0 0 20px 0;  
}  
.static-width {  
 width: 300px;  
 height: 100px;  
}  
.percentage-width {  
 width: 70%;  
 height: 50px;  
}  
Whatever width the parent element is, I want this to be 70% of that.

.inline-block {  
 width: 40%;  
 display: inline-block;  
}  
Will display like inline elements, but will keep the benefits of the box model, so they’ll be 40-40% wide of the parent element’s width.

Rounded corners:  
  
#main-content div {  
 background-color: #ccc;  
 margin: 0 0 20px 0;  
}  
.static-width {  
 width: 300px;  
 height: 100px;  
 border-radius: 10px;  
}  
specifies border radius of ten pxs on every corner. The higher the number the more curved the border is gonna look like.

.percentage-width {  
 width: 70%;  
 height: 50px;  
 border-radius: 5px 10px 20px 40px;  
}  
.inline-block {  
 width: 40%;  
 display: inline-block;  
}

.percentage-width {  
 width: 70%;  
 height: 50px;  
 border-radius: 10px 20px;  
}  
Would make top left 10px, bottom right 10px, top right 20px and bottom left 20px.

We can also draw a circle like this:  
.inline-block {  
 width: 100px;  
 height: 100px;  
 bottom-radius: 50px;  
}

Backgrounds:  
<div class=”static-width”></div>

.....

.static-width {  
 width: 300px;  
 height: 300px;  
 background-color: #606060;  
 background-image: url(„anakin.png”);  
 background-repeat: no-repeat;  
 background-position: center;  
 background-size: 200px;  
}

By default, the image repeats itself from left to the right and from top to the bottom.  
background-repeat: round; would make it fit into the grid, repeating itself around, it makes sure the image isn’t cut off at the edges.  
background-position: center; will align the image in the center of the element, both vertically and horizontally. You may also pass in percentage values or pixels instead of key words. First value means from the left, second means position from the top  
background-size: 200px; would resize the image to the given value.

Background Shorthand:  
we can combine a few of these properties into one line.  
.static-width {  
 width: 300px;  
 height: 300px;  
 background: url(„anakin.png”); no-repeat top center;  
 background-color: #606060;  
 background-size: 200px;  
}  
always add the background-color after the background shorthand. This way first, the browser will add the background-color underneath and then place the image on top, which will not overwrite the color.

Multiple backgrounds:  
<div id=”banner”></div>

.....

#banner {  
 background-color: #606060;  
 width: 100%;  
 height: 300px;  
 background-image: url(„https://ehaueh.jpg”), url(„http://ejrjererue.jpg”);  
 background-repeat: no-repeat, no-repeat;  
 background-position: center, top left;  
 background-size: 300px, 100%;  
}

Watch out for the comma between the urls! the image that comes first, will display on the top, the second will come next. The comma comes consistently into the background-repeat declaration, although here we could use the shorthand background-repeat: no-repeat; since both values are the same.

Pseudo-selector: módosítják a selector scope-ját

<https://css-tricks.com/useful-nth-child-recipies/>

Inline elemeknek nem lehet szélességet-magasságot állítani. Paddinget és bordert lehet állítani rájuk, margó csak horizontálisan fog működni.

bada55.io, pageruler, colorzilla