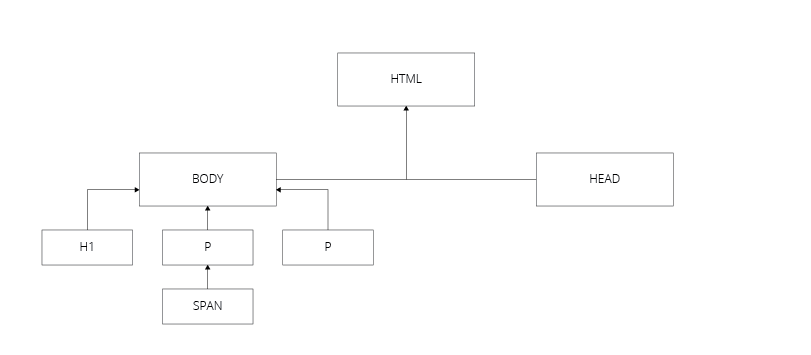
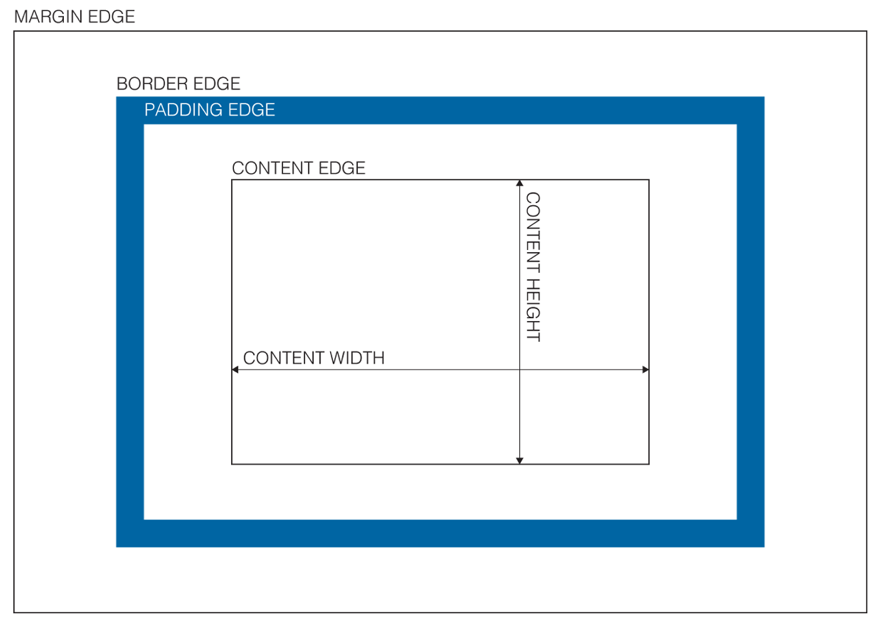
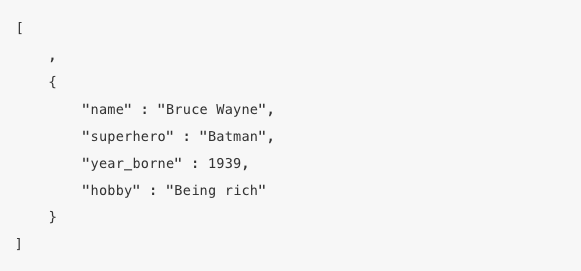
# Webteknologi – IT2805

* **Web architecture**
* Client-server architecture, browser is the client, server provides the files you see
* Browser is not the only way to connect, there are other ways than HTTP (HyperText transfer protocol)
* **Server**
* Software running on a computer, responds to request from users
* Web page and database exists on the server’s local file system
* Retrieves and transmits file to the client
* **Client**
* User can access the web through software called a browser (Chrome, Safari, Mozilla, etc..). The browser locates and displays information
* Server-client communication is done by an agreed transmission language (e.g HTTP)
* User requests web page through browser, which communicates with server. Browser waits for web page to be delivered (typically an HTML file), then renders the page.
* **Cloud**
* The cloud is a network of servers for the purpose of sharing resources
* **Network**
* Computers, both servers and clients, that are connected is a network
* Computers that are close (within a single department or building) make up a LAN (local area network)
* Computers covering a wide area (several buildings) is called WAN (wide area network)
* Largest WAN is the internet
* **The internet**
* Created by the Advances Research Project Agency (ARPA) and U.S department of defense for scientific and military communications. (ARPANET)
* Is a “network of networks”, made up of millions of computers, smart phones, PDAs, TVs, etc.
* The physical structure of the Internet uses fiber optic cables, satellites, phone lines and other telecommunications media
* TCP/IP
* Every computer and network on the Internet are using the same protocols.
* TCP/IP (Transmission Control Protocol/Internet Protocol)
* Was developed to tolerate unreliable sub-networks and the protocol guarantees proper transmission of data, since the physical network can’t
* Four layers:
  + Application Layer
    - Provides the ability to access the service of the other layers and defines protocols that applications use to exchange data
  + Transport layer
    - Responsible for making sure that complete messages are delivered
  + Network/Internet layer
    - Responsible for routing messages from one place to another
  + Physical layer
    - Responsible for actually translating the software message into a physical representation and putting them on the wire (or through air)
* Many different protocols on each layer:

|  |  |
| --- | --- |
| LAYER | PROTOCOL |
| Application | HTTP, telnet, FTP, email, VoIP |
| Transport | TCP, UDP |
| Network | IP |
| Physical | Ethernet, WiFi, ATM, X.25 |

* Domains and DNS
* Most internet hosts, or servers, have a domain name address in addition to an IP address
* A domain name “translates” an IP to letters, so that we can remember them easily.
  + Example:
    - NTNU.no is a domain, but what you are actually connecting to is “129.241.160.102” (try typing the IP into the address bar of your browser)
    - (You can find the IP for each web page by “pinging” them.   
      Windows: Open CMD and type “ping ntnu.no”  
      Mac: Open Network Utilities and go to ping. Type in ntnu.no)
* The domain name is made up of name levels so .no is Norway and ntnu is NTNU’s web page. Some enterprises have multiple servers with subdomains, such as idi.ntnu.no

|  |  |  |
| --- | --- | --- |
| Top level domain | Definition | For use by |
| .com | Commercial | Businesses |
| .edu | Education | Universities |
| .gov | Governments | U.S Federal government agencies |
| .int | International | Organizations established by international treaties |
| .no | Norway | Norwegian organizations |
| .mil | Military | U.S military |
| .io | British Indian Ocean Territory | British Indian Ocean Territory organisations and modern web companies |
| .net | Network | Network providers |
| .org | Organizations | Non-profit or miscellaneous organizations |

* The IP addresses are provided by Domain Name Server (DNS) computer’s “phone list” that map symbolic names to their IP
* Routing
* When a computer connects to the Internet, it is connected to a smaller network that is connected to the Internet’s backbone.
* The request data are broken into packets and travel across multiple networks before reassembled at their destination
* Dynamic routing – Routes are selected at the time of transmission, after considering current network conditions.
* World Wide Web (WWW)
* Many confuse the Internet with applications that work *over* the Internet. There are many of these, here are some examples:
  + Email
  + Telnet
  + File Transfer Protocol (FTP)
  + Internet Relay Chat (IRC)
  + The World Wide Web (WWW)
* Founded in 1989 CERN
* System of interconnected hypertext documents
* **Document Object Model (DOM)**
* Programming interface for HTML and XML document.
* Structured representation of the document, defines a way to access the objects from programs or scripts
* Structured group of nodes
* The HTML:  
    
  <html>  
   <head></head>  
   <body>  
   <h1Title</h1>  
   <p>  
   Some <a>linked</a> text  
   </p>  
   <p>  
   And some unlinked text  
   </p>  
   </body>  
  </html>  
    
  Will give the document tree:  
  
* **HTML**
* Describes the structure of a page
* Built up by tags, such as <body> <p> <div>
* At the top of the document you should include <!DOCTYPE html>, so the browser know it’s HTML5
* Everything that shows up on the page is within the <body> tag. Elements inside this tag is either inline or block
* Block elements -> Starts on a new line and takes up full width. Examples are div, h1-h6, p, form
* Inline does not start on a new line, takes up as much width as necessary. Examples: span, a, img
* Each tag has attributes. Example “src”. <img src=”image.png”>
* Some useful and basic HTML tags:
  + Headings: h1-h6
  + Paragraphs <p>
  + Bold: <b>
  + Italic: <i>
  + Superscript and subscript: <sup> and <sub>
  + Line break: <br> (no closing tag)
  + Link: <a href=”www.google.no”>This is a link to Google </a>
  + Lists: <ol> for ordered <ul> for unoredered. <li> for items:
    - <ol>  
       <li>List Item</li>  
      </ol>
  + Images: <img src=”....”>
  + Tables: <table><tr><th><td>
    - <table>  
       <tr>  
       <th>  
       <td>First head</td>  
       </th>  
       </tr>  
       <tr>  
       <td>Under first head</td>  
       </tr>  
      </table>
  + Comments: <!—This is a comment -->
  + <footer> <section> <header> <article> <input>
* **Site structure**
* URL
* URLs are used to link to another document or resource
* Precise location
* URL for webpages must start with “HTTP” or “HTTPS”
* An URL may look like <http://developer.mozilla.org/en-US/docs/Learn/>
* An URL is composed of different parts, some mandatory and others optional:   
  [http://www.example.com:80/path/to/myfile.html?key1=valu e1&key2=value2#SomewhereInTheDocument](http://www.example.com:80/path/to/myfile.html?key1=valu%20e1&key2=value2#SomewhereInTheDocument)
  + [Www.example.com](http://Www.example.com) is the domain of the URL
  + :80 is the port, that indicates the “gate” used to access resources on the web server (standard port for http)
  + /path/to/myfile.html is the path to the resource on the web server
  + ?key1=value&key2=value2 are extra parameters provided to the web server.
  + #somewhereInTheDocument is and anchor to another part of the resource itself
* GET and POST
* Consider the parameters in the URL (?key1=value…)
  + These are used with a method called GET.
* Two main methods when doing HTTP requests, GET and POST.
* If you want to request data, you use GET. If you want to submit data (such as a form), you use POST
* GET methods does **not** manipulate data on the server side, so a GET method should return the same every time.
* POST will update data on the server side, and thus changes the state of software.
* Example of GET in the address bar Is ?userid=5, this can for example be if you are logging in to a web page, and your user ID is 5.
* Properties of GET:
  + Can be cached
  + Can remain in the browsers history
  + Can be distributed and shared
  + Have length restrictions (because the URL has a max length)
  + Can me hacked, so don’t send sensitive data with GET
* POST requests are sent as a separate message, and not in the URL, they
  + Are **never** cached
  + Do **not** remain in the browsers history
  + Can**not** be bookmarked
  + Should be used for sensitive data
  + No max length whatsoever
* You specify if a form will use GET or POST as an attribute: <form method=”POST”>
* **CSS (the boring part of IT2805)**
* CSS -> Cascading Style Sheets
* Used to style pages
* Syntax:  
  Selector {  
   Property: value;  
  }
* Example:  
  P {  
   color: blue;  
  }
* Comments in CSS: /\* This is a comment \*/
* Three ways to style (in hierarchy order):
  + Inline (<p style=”color: blue;”>Text will be blue</p>)
  + Embedded (<style></style> in head)
  + External (external document, link with <link href=”soruce” type=”text/css” rel=”stylesheet”> in head)
* To style IDs, use #id { color: blue; }
* To style classes, use .class { color: blue; }
* Using \* { color: blue; } will select all elements on the page
* Child selector:
  + Li > a { color: blue; } will make all a elements ***inside*** any li element blue
  + Does work with ID and classes aswell #id > a { color: blue; }
* Descendant selector:
  + Matches an element that is a descendent of another specified element.
  + Example:
  + Given the HTML <div><p><a></a></p></div>
  + The CSS that will apply to the a tag is:  
      
    div a { color: blue; }  
      
    The child selector will not work here, because a is not a direct child of the div element. In other words:  
      
    div > a { color: blue; }   
      
    will not work, because the a is inside a p that is inside a div.
* Sibling selector:  
  h1+p { color: blue; }
  + Selects the first P element after a H1
* General sibling selector:  
  h1~p { color: blue; }
* Colors:
  + Can use HEX ( #FFFFFF for black), RGB( color: rgb(0, 0, 0) will be black) or color names (color: blue)
* Margin and padding
* The box model:  
  
* Use margin-left, margin-right, margin-top and margin-bottom (or just margin)
* Center an element with margin like this:  
  Margin-left: auto;  
  Margin-right: auto;  
  Width: 100%;
* Borders
* Can be styled using border-style, border-color and border-width
  + Can be:
    - None
    - Hidden
    - Dotted
    - Solid
    - Dashed
    - Solid
    - Double
* To style each side of the border differently:
  + Border-width: 10px, 2px, 20px, 4px;
  + Will give the top border a width of 10px, right 2px, bottom 20px and left 4px
* Text and fonts
* Font-family: Georgia, Times New Roman, serif
  + Use font Georgia if the browser supports it
  + Times New Roman is a fallback font in case the browser does not support Georgia
  + Serif is the “font style”, which is executed if Times New Roman fails
* Font-size for font size (px, em or %)
* Line-height for the line height (150% = 1.5)
* Color: blue will give the text a blue color
* Pseudo-classes
* A keyword added to the selector, for example :hover, :visited, :checked
* Example:
  + A:hover{color: blue;}
  + When you hover over the link, the text will turn blue
* A little more useful css:
  + List-style-type, to change the black circles into squares, disc, non-filled circles, none etc..
  + Visibility: hidden/visible (will still be there if hidden, as an invisible block)
  + Z-index to put an element in front of another
  + Position
    - Fixed
    - Static (default)
    - Relative
    - Absolute
  + Float (for example to get text beside an image)
* Inheritance
* Any child node will inherit the parent node’s css (e.g p inside a div, will inherit the div’s css), unless other is specified. Can have div{color: blue} and p{color: red}
* More pseudo-classes:
  + :nth-child(), picks the nth child:
    - li:nth-child(2n+1), child 1, child 3, child 5, etc…
    - li:nth-child(2n)
    - li:nth-child(odd)
  + :last-child(), pretty self-explaining
  + :first-letter, :last-letter
  + :required
  + :focus
  + :checked
  + :disabled, :enabled
  + and soooo much more
* **Web and multimedia**
* Multimedia = images, video, audio
* Media files should not be larger than necessary (think about loading time and cellular data etc..)
* Images
* Should be fitting the context
* Right permissions to use it
* The img tag should have the attributes “src” and “alt”
  + Alt is for text if the browser can’t display the image
  + The img should look something like this:  
    <img src=”path/to/file” alt=”alternative text”>
* You can also style images with width and height
* Img is inline
* Image file formats
  + Not all formats are relevant for the web
  + JPEG
    - For images with many different colors.
    - Lossy file format, meaning there will be a loss in quality when compressing it
    - Does **not** support transparency
  + PNG
    - Lossless format, meaning the quality isn’t reduces even when compressed
    - Works for images with few colors
    - Can be 100% transparent
  + GIF
    - Lossless format
    - Max 256 colors
    - Animations
    - Transparent
  + SVG
    - Vector format
    - Not stored as pixels, so they can be resized without losing any information/quality
    - Ideal for logos
* Video
* Useful attributes:
  + Src
  + Controls
  + Autoplay
  + Height
  + Width
  + Loop
  + Muted
  + Poster
* Audio
* Useful attributes:
  + Src
  + Controls
  + Autoplay
  + Loop
  + Muted
* **Forms and validation**
* Forms allows users to send data to a server through a web site
  + Example sign up/in
* Consist of one or more input fields
* Submit button
* <form> tag
* Block element
* Action and method attribute
* Form elements can be used outside the form element
  + If you do this, you have to customize the behavior with JavaScript
* Most common form element is the <input> tag
  + Use the “type” attribute to specify the type of input
  + Examples are:
    - Text (default)
    - Number
    - Email
    - Date
* Use the <select> tag for dropdown options
* Form validation:
  + Should validate at the server side
  + Should also validate before the user sends it
    - This can be done by using the correct type attribute
  + To give the user feedback on where the error is, use the css pseudo-classes :valid and :invalid
* Then something about form styling, but eh
* **Javascript (the fun part :D)**
* Can be written embedded inside the HTML document or external
* Place JS in <script> tags if embedded
* Browser reads HTML from top to bottom, so if you want some JS to load before the page, place it in the <head> tag
* Comments in JS:
  + Single line //This is a single line comment
  + Multiple lines /\*  
     This is   
     a multiline  
     comment  
     \*/
* Data types in JS:
  + Strings, “This is a string”
    - Can add strings to each other like this:
    - “Hello, my name is “ + “Sander” will be “Hello, my name is Sander”
  + Numbers (or integers), 1 <- That’s an integer
    - Can use mathematical operations on them
      * +, -, \*, /, %
    - They can also be Infinity, -Infinity and NaN (Not a Number)
  + Boolean (true/false) (1 = true, 0 = false)
* Logical operators in JS:
  + < (less than)
  + <= (less than or equal to)
  + > (greater than)
  + >= (greater than or equal to)
  + == (equal)
  + === (precise equal)
  + != (not equal)
  + !== (not precisely equal)
  + && (and)
  + II (or)
* 4 < 4 === False
* 4 <= 4 === True
* “2” == 2 === True
* “2” === 2 === False
* 5 != 4 === True
* (5 === 6) && (8===8) ==== False
* Variables
* Variables hold values to later use
* To declare a variable write: var myVariable;
* To declare and assign a value: var myVariable = “My variable”;
  + A variable of string type
* You can also declare by using “const” and “let”
  + Use const if the variable won’t change
  + Use let where you would use var
* To say something doesn’t exist:
  + Null or Undefined
* Control structures:
  + If (variableName === “30” ){ do something}else{do something else}
  + You check if something is true, and then does something if it is
* Arrays
  + Same as python lists (ish)
  + var emptyArray = []
  + var myArray = [“Hei”, 2, “på”, True, “deg”]
  + Arrays is a list of elements, can hold any datatype
  + 0 indexed, meaning the first element is at index 0:
    - myArray[0] = “Hei”
    - myArray[1] = 2
* Objects
  + Same as python dictionaries (ish)
  + Var emptyObject = {}
  + Var myPersonObject = {name: “Sander”, age: 19, single: True}
  + myPersonObject[“name”] = “Sander”
    - Note: when accessing, use “”, when declaring (inside the {}, do not use “”)
  + To add a new key: myPersonObject.hometown = “Trondheim”
* Loops:
  + Execute same code multiple times
  + For loop
    - for(let I = 0; I < 10; i++) {console.log(i);}
    - for as long as i is less than 10, console.log(i), and update I by one each time
    - General:
      * For(initiate; test; increment){do something}
  + While loop:
    - Let I = 0; while(I < 10){do something; i++;}
    - While I < 10, do something, increment I each time
    - General:
      * Inisiate; while (test) {do something; increment}
    - REMEMBER to increment, else the loop will loop forever
  + Break and continue:
    - The break keyword breaks the loop, continue “restarts” the loop, the loop ignores the rest of the code and continues at the top of the loop
* Functions:
  + Useful if you have code to run multiple times, on different places in your code. It “stores” code.
  + Declaration:
    - Var square = function(x){return x\*x}
    - or function square(x){return x\*x}
    - x is called a parameter (works as a variable in the function)
  + Call:
    - Square(10), will return 10\*10 = 100
* JSON:
  + A widely used data storage and communication format for the web.
  + Can contain anything a JS object can
  + Written like this:  
    
  + JS JSON functions:
    - JSON.stringify() and JSON.parse()
    - If you have an object and use JSON.stringify(object), you convert the object to JSON string

There are also some more advanced JS, such as how to filter an array, mapping them, sorting them etc.. Also something about scope, it’s not that hard, just remember local and global variables. Local are only accessible within a function, global are accessible everywhere.

How to manipulate the DOM?

Let firstP = document.getElementById(“firstP”); //Reference to the html element with id firstP

firstP.innerHTML = “This text will replace the text already in the P element”;

firstP.style.color = “Blue”; //Changes the color of the paragraph to blue

let groceries = document.getElemensByTagName(“li”); //This will be an array containing all li elements

groceries[0].innerText = “Juice”;

You can also use document.getElementsByClassName(“classNameHere”);

There are also something about .removeChild’n stuff, but I don’t think it’s very important for the exam.

Events can be useful. You have onclick, onkeydown, onkeyup, onmouseover, onscroll, etc… Basically any event you can think of. Example:

Let img = document.getElementById(“img”);

Img.onclick = function (){

If (img.src === “1.png”){

Img.src = “2.png”;

}

else{

img.src = “1.png”;

}

}

this will change the src of an image when the img is clicked.

Other useful JS functions are:

* Prompt(“Enter age:”) for user input
* Alert(“ERROR!!!!!!!!!!!!”) for alert window