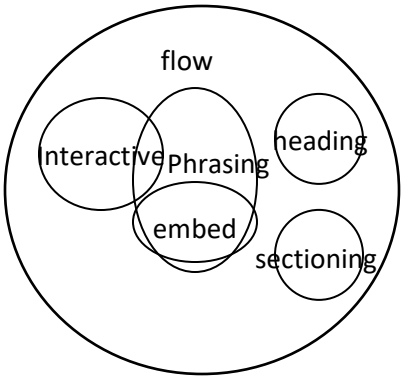
Web Systems and Technologies (midterm notes)

- HTML (Hypertext Mark Up Language)
    - to create web pages
        - document; contains some information
            - that is formatted in a special way
            - several aspects
    - divide into paragraph (eg. List, Contains, Structure)
    - web page cover by HTML
        - structure and content
            - statistics, storing
            - markup the structure and the content
        - aesthetics/presentational aspects
            - not modern practice
            - not use in today's generation
            - (eg <h1> Hello </h1>)
        - behavioral aspect
            - earlier age – static resource; look at the information, done.
            - today – there is interaction
    - CSWS (Client-Side Web Scripting)
    - JS (Java Scripting)
    - Tim Berners-Lee
        - father of the web
    - HTML Versions
        - HTML 1.0 – first version
        - HTML 2.0 – embodied to a technical document
            - RFC 1866 Nov 1995 – W3C (world wide web consortium) (same year) : organized by IETF (internet engineering task force)
        - HTML 3.0 – W3C Recommendation
        - HTML 3.2
        - HTML 4.0 – December 1997
        - HTML 4.01 – December 24 1999
            - 3 Different Variant (strict version)
                - Strict – mix of presentation and structure
                - Transitional – transition to new practice
                - Frameset – multiple frames (depreciated)
            - stop HTML and introduce XHTML (extensible hypertext markup language)
        - HTML 5 – 2014
            - XMTML and flexible syntax
        - HTML 5.1 – November 2016
            - specs
            - Ian Hickson (started)
        - HTML 5.2 – ongoing
        - XHTML 1.0 – January 26 2000

- reformulation of HTML in XHTML
    - markup data not web pages
    - rules are very strict
- deprecated – don't use it anymore
- rules..
    - <input disabled>
        - values
    - have beginning and ending
    - case sensitive – limits the connect
    - documents must be well form
    - camel case only (non case sensitive)
    - attribute values must always be open
    - end tag is require
- WHATWG (web hypertext application technology working group)
    - 2005 – 2006
    - Ian Hickson – works for google
    - started the new version of HTML
    - 2012 issue to standardize as root
- typical structure of a webpage (empty element)

```
<!DOCTYPE html>
<html lang = "en"> - html element, root
<head> - start of the CHILDREN
    <meta charset = "UTF-8"> - empty: no content; information of data
    <title> </title> - author, keyword, title of the document
</head> - end of the CHILDREN
<body> - actual content to be seen by the user/client
<p>
<a href = "link"> - <a> beginning/start tag – href attribute/tag name
"link" content/value - </a> end tag
</a>
</p>
<img src = "" alt = "">
<!—comment -- > - to insert a comment
</body>
</html>
```

- p element (paragraph)



        - heading – h1, h2, h3
        - sectioning
        - phrasing – u, var, bdo, canvas, video
    - flow content
    - content model : phrasing content
    - tag omission in text/html : A <p> element's end tag
    - content attributes: global attributes – lang, id, hidden, dir
    - DOM interface : interface HTMLParagraphElement: HTMLElement ();
- img element (image)
    - embedded content
    - content model: nothing (empty element)
    - tag omission: empty
    - content attributes: alt, width, height
    - DOM Interfaces:
- HTML Elements:
    - html (root element)
    - head (child)
        - title – title
        - base – relative
        - link – different resource (eg stylesheet)
        - meta – other meta info (eg keywords, author tool)
        - style – embedding
    - body (child)
        - article, aside, nav, section
        - header, footer
        - main (main content)
        - address (contact information)
        - div (generic section content)
        - h1, h2, h3, h4, h5, h6 – sub-heading
            - main heading
        - p (paragraph)
        - hr (horizontal line) – hematic break: theme break
        - pre (preformed text) – preserved: white space, new line
        - blockquote: quotation (q) , citation (cite)
        - list: ol (ordered), ul (unordered), li (list); (eg <ol start = 10>
        - description list (dl) : dt (term), dd (actual)
        - figure: figcaption
        - a (anchor)
        - em, strong (emphasize), small, s (side comment), (obsolete text)
        - cite, q
        - defining instant: dfn, abbr (abbreviation)
        - ruby, rb, rt, rtc, rp
        - data, time (information/code)
        - code, var, samp, kbd (embedding computer code)

- superscript (sup), subscript (sub)
- i (italize), b (bold), u (underline) – depreciated element; 401 strict, mark
- by directional text: bdi, bdo
- text level division: span
- break opportunity: br, wbr
- news, documents edit: ins, del
- table: table, caption, colgroup, col, +head, +body, +foot, tr, th, td
- form: label
- math: svg, canvas (eg snake game)
- script, no script
- template

- HTML/XHTML Stylesheets
  - author styles – author of document/web development team
    - external styles – link across multiple html

```
<head>
    <link rel = “stylesheet” types = “text/css” href = “style.css”>
    // <media = print (css3), screen (css3), tv, brail, projection,
    speech (css3)>
    // <title = large, normal, small font>
    // <link rel = “stylesheet”/”alternate stylesheet”>
```

    - embedded stylesheet – single page website

```
<style type = “text/css” media = “screen and (min-width:500px;)
    body{width:90%;}
</style>
```

    - inline stules

```
<p style(attribute) = “border:1px solid green; color:blue;”/>
</head>
```

  - user style – provided by the area; viewing the page
  - user agent styles (example default css 2.1 stylesheet for html4)
    - browser itself (default css stylesheet)
    - ResetCSS – Xss stylesheet that can link to everything

- HTML Processor
  - HAML (Ruby), Markdown, Slim, Pug

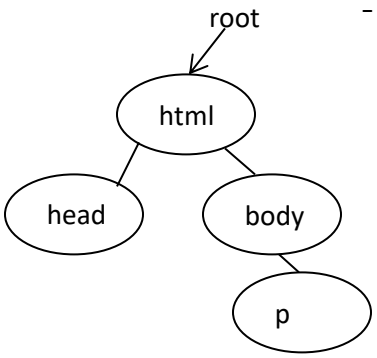
Ex.

```
%body                          <body>
%div                           <div>
%p                             <p> hello world </p>
    Hello world                </div> </body>
* - var = “my variable” -> %h1 = var
* %p.abc content          <p id = abc > content </p>
```

- CSS (Cascading Style Sheet)
    - structural markup documents
    - language used to specify the presentation aspects (eg layout and formatting) of structurally marked up documents
    - developed by Hakon Wium Lie (CHSS: html cascading style sheet) and Bert Bos (SSP: stream-based style sheet proposal)
    - versions:
        - monolife – everything is covered in the specs
            - CSS1 – December 17 1996 revised April 11 2008
            - CSS 2.1 – June 07 2011
        - CSS3 – recommended; individual document defining each module
        - CSS Preprocessors, CSS Frameworks (eg $color: red -> body{color=$color;})
            - SASS (better ways for CSS), LESS, 960 Grid System, Bootstrap, Foundation, Materialize (formatting for tabs, tables), etc.
                - material design: concept introduce by Google
    - CSS Statements
        - At – Rules
            - @charset
            - @namespace
            - @import
            - @document – specific page or with given url or domain
            - @fontface – downloadable files
            - @keyframe – animation
            - @media – responsive design
            - @page
    - CSS Rule Sets (aka CSS Rules, Style Rules)
        - selector followed by brace-enclosed declaration block

        p (selector){ - target particular element
            color: #f0o;
        }
        /* ---- */ - block comment
        // - make it invalid/ improper comment
    - CSS Selector
        - Selector
            - elements in the document tree are matched
            - body ail# x>p-y[title] + span: +-before
            - condition in a CSS rule
            - match elements called the subject of the selector
        - Selector syntax
            - one or more sequences of simple selector
            - ex. body(simple) p.xyz (selector sequence) + h1[title]::before (pseudo element)
        - sequence of simple selectors
            - chain of simple selector not separated by

- ex. h1, h2, h3 – group of selector
- simple selectors
  - type selector – name of elements in html
  - universal selector – represented by *; target everything
  - attribute selector – eg p[title] class = 'x y'
    - [attr]
    - [attr = value] – specific value
    - [attr ~= value] – one value out of set of several values
    - [attr |= value] – language attribute; value = *something*
    - [attr ^= value] – and
    - [attr $= value] – dollar
    - [attr *= value] – anywhere
  - class selector – eg class = 'x y' -> p.x
  - id selector – eg id = 'id' -> #id
  - pseudo-class
  - dynamic pseudo-class
    - link pseudo-classes
      - -:link
      - -:visited
    - user action pseudo-class
      - -:hover
      - -:active
      - -:focus (tab key)
    - target pseudo-class
      - -:target eg _#x (url)
      - a href = '#t1' – internal link
      - p:target{
        display: none || block;
        }
    - language pseudo class
      - -:lang()
    - UI element states pseudo class
      - -:enabled
      - -:disabled
      - -:checked
      - -:indeterminate
    - structural pseudo-classes
      - -:root
      - -:first-child (children)
      - -:last-child (children)
      - -:only-child (children)
      - -:nth-child(n)
      - -:nth-last-child()
      - -:first-of-type



-:last-of-type
-:only-of-type
-:nth-of-type()
-:nth-last-of-type()
-:empty (eg u1:empty{width:100px height:100px}
- negation pseudo-class
-:not() – (eg body:not(h1){ - type selector/simple selector
color: blue;
}
- combinators
- descendant combinatory
- whitespace – space, tab, linefeed, carriage, return, form feed (eg body p{ //paragraph descendant of body
- child combinatory (eg body > p { //levels below
- sibling combinatory
- adjacent sibling combinator (+)
(eg h1+p //immediate follows
- general sibling combinatory (~)
(eg h1~p //paragraph follows h1
- pseudo elements – format just part of it
- ::first-letter, :first-letter
(eg h1::first-letter{font size: 2em;}
- ::first-line, :first-line
- ::before, :before – introduce content before
(eg h1::before{content:"topic";}
- ::after, :after – introduce content after element
- cascading – all elements gets combined

- CSS Rule Precedence
- by origin and importance
- user agent important declarations
- user important declarations - override the author
- author important declarations
- author normal declarations
- user normal declarations
- user agent normal declarations – (eg color: red; important;)
- transition declarations – change the value of the property from one to another
- import override declaration – use javascript to import
- animation declarations
- by specificity - the name specific the higher the importance
- inline style – put it on the element itself
- number of ID selectors
(eg id = x => div # x > p# y (important)
<p id = y #x>p (lesser important)

- number of class selectors, attribute selectors and pseudo-classes
- number of type selectors (p, h1) and pseudo-elements (before, after)
(eg body p (more specific) ; p (lesser)

- by order

- CSS Declarations
  - properties
    - shorthand properties
    - vendor – specific extensions aka vendor prefixes – experimental
  - values
    - keywords, numbers, dimensions (x)
      - integers and reals in decimal notation
    - dimensions
      - length, angle, duration, frequency, resolution
      - length units
        - font-relative – em, ev, ch, rem
        - viewport – percentage – vw, vh, vmin, vmax
        - absolute lengths – cm, mm, q, in, pt, pc, px
      - angle units – dev, grad, rad, turn
      - duration units – s, ms
      - frequency units – hz, khz
      - resolution units – dpi, dpem, dppx
    - percentage – (eg 10%)
    - URLs and URIs – (eg url(_____)),hsl() – hue, saturation, lightness
    - colors (eg red; #rrggbb; #rgb; #rgba – 0(transparent) -> 1), hsl (0, 100%)
      100 – white; 0 – black; 50 - normal
    - strings
    - functions – calc(), attr(), counter(), counters(), linear-gradient()
    ~ background-color: red; (inherit, initial, unset)
      - inheritance - not inheritance

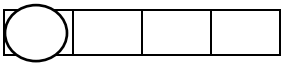
- CSS Preprocessors, CSS Frameworks
  - extend the syntactical capability of CSS
  - examples:
    - SASS – preprocessor; convert it to CSS
      - syntactically awesome stylesheet; new SCSS
        - eg div{ p{ Color:red; }} => //color:blue; div p{ color:red; } //color: color;}
        - eg @mixin pm{ padding = 0; margin = 0;} => div{ @ include pm }
        - eg @ function twice($v){ @return 2*$v;}
        - eg @ for $i from 1 through 6{

```
H#{$i}{
font-size: 30px – ($i-1) * 3;}}
        ||
h1{
        font-size: 30px;}
h2{
        font-size: 27px;}
```

- nested part :

```
font:{
  family: seriff;
  size: twice(20(int; dimension)em);}
```

- build in GUI (Graphic User Interface)
- eg div => div p{ color:red; }
  p
  color: red
- LESS – works on javascript; node.js
- SCSS
- Stylus
- PostCSS
- 960 Grid Systems
- Bootstrap
- Foundation
- Materialize

- <span class = 'cell-r1-c1' <-  -cell-r1-c1{___}

- CSS Framework
  - bootstrap (bootstrap.com)

  click
  <span class = 'btn btn – (xs, (sm-small), lg (large), md, xl)
  <table class = 'table table – hover table – stri </table>
  <ul class = 'nav (nav-pills, active, nav-tabs)'>
  <img src = "___.jpg" alt="">

- Java Script
    - imperative programming language
    - sequence of statement
    - sequence of instructions that modify
    - dynamically type language
        - never designate types
    - defacto programming language for the client side
    - eg var x=100;//number
        x = hello; //then become a string
        function max (x,y){ //declaring a function to return type
        valid //max(5,10) || max('hi', 'hello') || max (0, true)
    - DOM not part of Javascript
    - keywords:
        - var
        - let – only available within the block
        - const – constant declaration; cant re-asign it to other
    - eg <script> //top level code
        var x=100; //property
        * let x = 100; //not get an error, just create it
        function f(){ //function level code
            var x = 200; //local variable; doesn't exist outside of the function
                - //function scope variable
        } let x = 200; //only available in this part only; in this block only
            - block local variables

    - Java
        - safe, very rigid
    - JavaScript
        - flexible, know what you are doing

    - Client-side Java Script
        - MDN – Mozilla Developer Network
        - ECMAScript (Standard)
        - 3 Kinds of putting the JavaScript
            - <- externally link -> (there's an defer attribute)
            - <- embedded script -> (execute as it is encountered)
            - <- inline script ->
        ~ parse the engine starting from the top and fetch the .js (compile -> execute)
        ~ put code on the script
            - top – level tool

```
<script>
    console.log ('embedded script…');
    function callMe(){
        console.log ('you called…');
    }
</script>
```

<buttononclick = 'console.log('embedded script…'); callMe()'>
- defer attribute
    - from the top and when it gets through it (|) will compile -> execute
    - start fetching and not wait until execute fully done
    - eg ~~ hello
    - <script> defer type = 'text/javascript' src = 'script.js'>
- async attribute
    - continue rendering the document, but when it is available, it executes and compile it.
    - eg ~~~ hello
    - <script type = 'text/javascript' src = 'script.js'> //up or down
    - <noscript> no scripting support </noscript> ~no support for scripting
- pre-defined object
    - window (global object) [window.screen.width]
        - navigator – vendor, appversion
        - screen – width, availHeight
        - document – getElementById, childNodes
            Examples
                - document.getElementById('h')
                - h.innerText = 'hi'
                - h.innerHTML = 'hi there '
                - h.textContent
                -h.title
                -h.lang
                -h['lang'] = 'en.us'
                - h.data-extra
                    - not valid character in name of javascript
            - get/set – on normal attribute
                - h.setAttribute ('data-extra', 'value')
                    - use if not part of …
                - h.get Attribute ('data-extra')
                - h.style = 'color: blue'
                    - create an inline style
                    - available (all) on the style attribute
                    - all values (if properties) must assign as a string ('hi')
                - h.style.fontSize = '3em'
            - attribute name
                - h.className = 'green-text'
                - document.querySelector('p')
                    - return the first element that matches it
                - document.querySelectorAll('p') /('p:first-child')/

('p:first-of-type') / ('h1+p')
- first's p - follow h1
- s.querySelectorAll('p')
- all with 's' element
- s.matches ('p') //=false
- elements not paragraph
- doctype: document.childNodes
document.children //=html
document.head.childNodes
//=title, style, text

- sandbox
  - limited environment to offer
  - run the browser
- ctrl + u
  - look for the script of the certain site in the browser
- window (DOM Tree (document))
- document object model
  - HTML/XHTML
  - type domination
- DOM3
  - attr = text, entityReference
  - document = element, comment
  - element = element
-//interface document: (extends/implements) Node{
  * w3c.dom (check for more information)
- Node
  - primary datatype for the …
- Constant
  - documentNode
  - document.type.node
  * - nodeName = #document
  * - nodeValue = null
  * - childNodes = text, em, text
    h1 – document.createElement('h1')
    * <h1>/<h1>
    t1 – document.createTextNode('text')
    * "text"
    h1 – appendChild <t1>
    * <text>
    h1
    * <h1> text </h1>
  - document.body.insertBefore <h1 (replacement), p (one who will going to be replace, child of another element)> //insert before the last one
    <h1> text </h1>

* - nodeType = (“number)
* - parentNode
- elementNode
- notationNode
- attributeNode
* - querySelector = one node
* - querySelectorAll = all nodes
- type of nodes
    - insertBefore
    - replaceChild
    - removeChild
    -appendChild
- document.importNode; document.adoptNode – transfer it
    - dealing with more than one document
    - copy to the new document and have duplicate it
- document.firstElementChild
    - came from the newest DOM – DOM 4
    - not all browser support this
    - still experimental and might be removed yet

- Browser Model Object Implied

<p innerHTML = ‘hi there

- Datatype
    - simple/primitive
        - boolean
            - v = true j>v=false || >typeOf v >Boolean
            ‘truthy’ – not really true but it is true
            ‘falsy’ – not really false but false
            - 0 = false; other # = true (Number); anything = true; empty = false (String)
            - undefined values = false; null = false
            - eg if(true){ console.log(‘yes’);}
        - numbers
            varn = 1 ; varn=1.2 – typeOf n||m = number
            0b101 = 5(binary), 0101 = 65(octal), 1c5 = 100000(hexadecimal)
        - strings
            ‘c’ = “c” || “c” = “c” ; “can’t” = “can\’t”
            - backcode – ‘jhdkl’ = “jhdkl”
            - eg i.innerHTML = ‘ jklm ’
        - undefined
            - anything that is undefined
        - null
    - structured/reference (standard object)

- array, Boolean, Date, Error, Function, JSON, Math, Number, Object, RegExp, String, Map, Set, WeakMap, WeakSet
- Boolean
    Eg b=true || b1=newBoolean(true)
    - type of b “Boolean” - type of b1 “Object”
    Number.Max.value = 1.7976977
- typeOf = Object //everything
- instanceOf = number/Boolean //exact representation
- Math(object)
    Eg Math.random(), Math.pi, Math.E

- Expressions and Operators
  - Javascript’s operators
    - instanceOf, typeOf, new, this, theoperator, precendence
- Function
  - procedural abstraction
    - name with a block of code
    - eg Function sayHello(){ //name of function
      Console.log (“hello”); - defined property that is executable
      }
    //sayHello -> look at the content //sayHello() -> see the output

NO:
DATE:

(eg2) function saySomething(something) { // print an argument
console.log(something); // something = "any string"
}

(eg3) function greater(v1, v2){ // return a value / undefined
if (v1>v2){
return v1;
} else if (v2>v1){
return v2; }}

explicit conversion // depricated in javascript

(eg4) function fn(arg){ // pass one argument
switch(arg){ // print fn(1) → 100 ; fn(2) → 'hello'
case 0 : return true; case 1 : return 100; }}
return as an argument

(eg5) function expression
(function(a,b){return a+b;})(10,20);
↳ function ↳ expression

var add = new function('a,b', 'return a+b');
↳ argument ↳ constructor ↳ body

(eg6) Arrow syntax (Fat arrow syntax)
var multiply = (a,b) => {return a*b;} // a/b //divide
↳ function ↳ return

var product = multiply(10,20); // var 0 = () => 0;

(eg7) recursion (factorial of number n = n! = n * (n-1)!)

function can be recursive

function factorial(n){
if (n<0){
throw "invalid argument";
} else if (n==0){
return 1;
} else{
return n * factorial(n-1);
}}

Spring Leaf

- Arrays
    - collection of things
    - Array Constructor: //array with empty elements/array
        var emptyArray = new Array(); -> (5,10,15); (‘5’);(String ‘5’) (5); - 5 elements array
    - Array Literal Syntax
        var alsoEmptyArray = [];-> [5,10,15]; [‘5’];
        int[] a = new int[10] – a only have 10 elements
            a.length; - 0-9 only
        - trancades/throws away – when assign is lower than element
        - var mixedElementType – [10, true, ‘hello’, new Date()];
        - var matrix = new Array[new Array(10,20); new Array(20,30);]
        - table (matrix) – to become a table
        - var multiDimArray {[‘apple’, ‘banana’], [4,5,6], []}
            array destructing
        - var array = [1,2,3,4,5]; 1=a, 2=b, 3=c, 4=d, 5=e
        - var [a,b,c,d,e] = array;
            array indices can be non-contiguous
        - var array = [1,2,3,4,5];
            array[10] = 10;

            for(let i in array){
            console.log(I, array[i]);}
    - Method = Arrays
        - Mutator Method – modifies the target array
            - Array.prototype.copyWithin()
            - Array.prototype.fill()
            - Array.prototype.pop() – removes last and return that element
            - Array.prototype.push() – adds one element to the end and returns new length in the array
            - Array.prototype.reverse() – 1st become last; last become 1st
            - Array.prototype.shift() – remove 1st and return that element
            - Array.prototype.sort() – sort and return the array
            - Array.prototype.splice() – add/remove element
            - Array.prototype.unshift()
        - Access Method – method do not modify the array and return some representation
            - Array.prototype.indexOf()
            - Array.prototype.join()
            - Array.prototype.lastIndexOf()
            - Array.prototype.concat()
            - Array.prototype.slice() – return the portion
            - Array.prototype.toSource() – deprecated
            - Array.prototype.toLocalString()
        - Iteration Method – perform operations in methods
            - Array.prototype.filter() – create new array and filter function

- Array.prototype.find() – returns found value in the array
- Array.prototype.entries()
- Array.prototype.findIndex()
- Array.prototype.forEach() – return each element
- Array.prototype.keys() – all the indexes
- Array.prototype.every()
- Array.prototype.value() – all the value
- Array.prototype.reduce() – 0 -> 100
- Array.prototype.reduceRight – 100 ->0
- Array.prototype.some()
- Array.prototype.[@@iterators]() – create own iterator

- Objects //prototype base
  - var emptyObj = new Object();
  - var alsoEmptyObject = {};
- Constructor Functions
  - Function Person (name, age){
    this.name = name;
    this.age = age;
    this.speak = function(){
    console.log('$(this.name)');}
  - p1 = new Person ('a', 1)
  - Person {name = 'a', age:1}