# Internetprogrammering
# DD1386

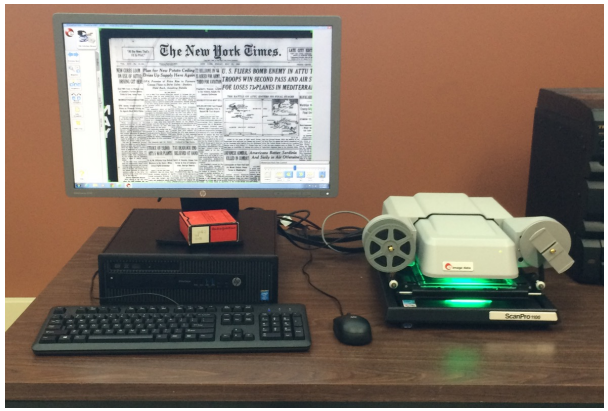
Föreläsning 4

# Innehåll

- (X)HTML
- CSS
- JAVSCRIPT
  - Variabler
  - Konstanter
  - Operatorerna ==, ===
  - Automatisk Typ konvertering
  - Array
  - If-satser
  - Loop
  - Funktioner
  - Closure
  - Högre ordningens funktion
  - Alert, log och Prompt
  - Object, Array, class
  - jason

# HTML

- **H**yper**T**ext **M**arkup **L**anguage
- **HYPERTEXT**: text med länk till andra text
  - 1945: microfiche, Vannevar Bush




  - 1989: Tim Berners-Lee, hypertext för internet

# HTML

- **Hyper Text:** text med inbyggda kopplingar till annan information (länk till annan information)
- **Markup Language**: Ett språk bestående av textkod som är instruktioner till ett program som beskriver hur ett visst dokuments olika delar som text och bild ska presenteras för användaren.

- Textkoden består av: HTML-element

```
<b>H</b>yper <b>T</b>ext
```

**H**yper **T**ext

# HTML taggar

- HTML element kan även ha attribut
  – <tag attribut1=”värde1” attribut2=“värde2” ....

- Taggar kan innehålla andra taggar och vara nästlade

# Några HTML-taggar

| Element | Beskrivning |
|---|---|
| <html> | Html-sida |
| <head> | Huvud-info |
| <Htle> | titel |
| <body> | innehåll |
| <h1>,..,<h6> | rubrik |
| <img source=””> | bild |
| <form> <input> | Webb-formulär |
| <table> <tr> <td> | Tabell, rad, kolumn |

| Element | Beskrivning |
|---|---|
|  | Gruppera element i felra rader |
|  | Gruppera element i en rad |
| <p> | Ny paragraf |
| <br> | radbrytning |
| <script> | Skript språk |
| <ol> | Ordnad lista |
| <ul> | Oordnad lista |

# Generella attribut

- HTML-Element har attribut som är specifika för varje element men några generella finns också
- Dessa gäller ej taggarna `base, head, html, meta, param, scr`ipt och title.
  - class (formateringsklass)
  - id (referens)
  - style (css)

# HEAD

- HEAD kan innehåller följande sub-element.
- <base href="http://www.csc.kth.se/~stene/" >
  - Prefix-url för alla länkar på sidan
- <link href="special.css" rel="stylesheet" type="text/css">
  - Definierar relation mellan två dokument, ofta för att inkludera externt stylesheet
- <style type="text/css">
  - - alltid ”type=text/css”. För att inkludera CSS information
- <script type="text/javascript">
  - Attributet ”type” anger MIME type t e x ”text/javascript” eller ”text/vbscript”. Ett externt dokument kan inkluderas med attributet ”src”
- <title>
  - Dokumentets titel
- <meta>
  - Information till sökmotorer m h a attributet ”name” men även HTTP-header information m.h.a attributet ”http-equiv”

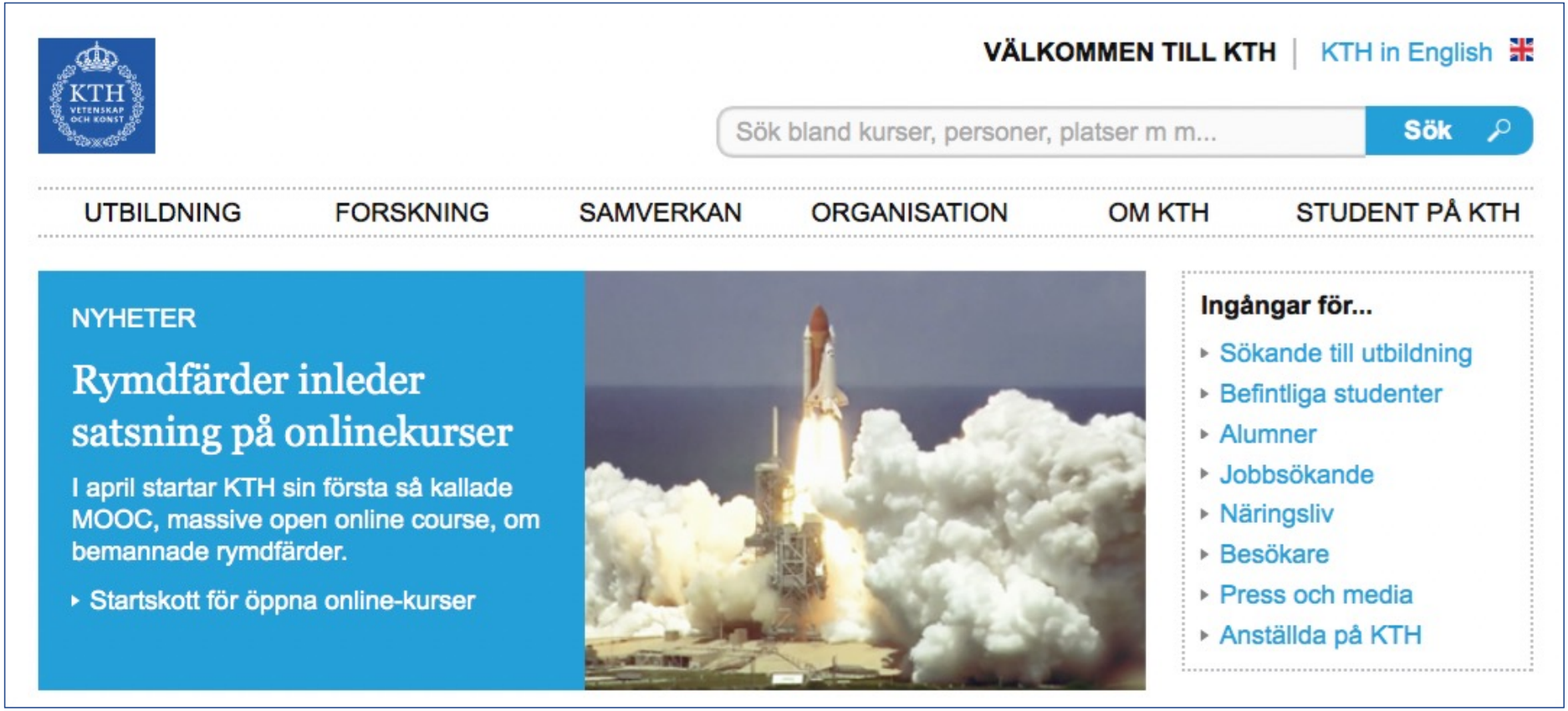
# Formulär & DOM

- I <form> taggen kan man lägga vilka presentationstaggar som helst, men den viktigaste är att <input> taggen inuti formuläret.
- <form name=”kunddata” method=”GET” action=”order.php”>.....</form>

- Alla taggar kan refereras via DOM-strukturen

| | |
|---|---|
| window.document.forms[0].efternamn.value | Värdet till variabeln efternamn i första formuläret |
| window.document.location.href | url-adressen till aktuell sida |
| window.history.length | Storlek av histoy |

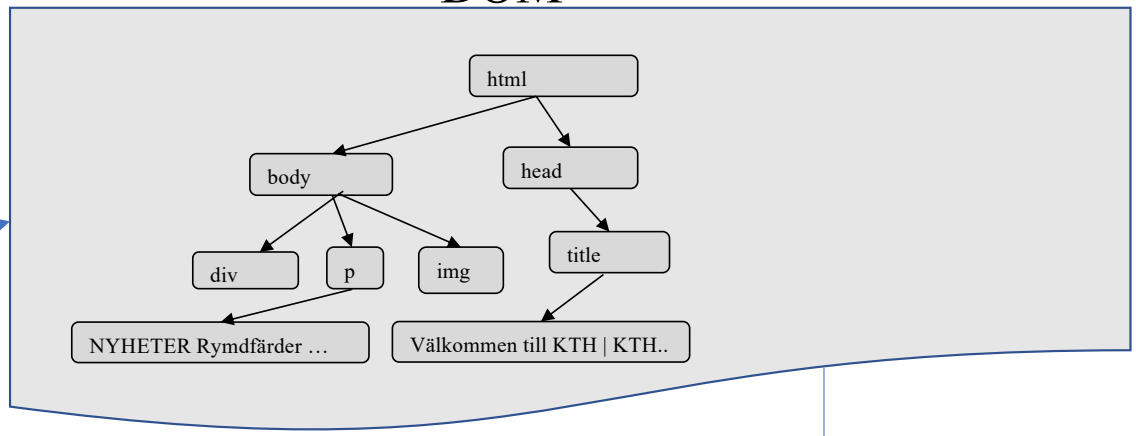
# Hos webbläsaren



(x)HTML

```
<html>
<head>
<link>
<script>…
<img>
…
<body>
```

DOM

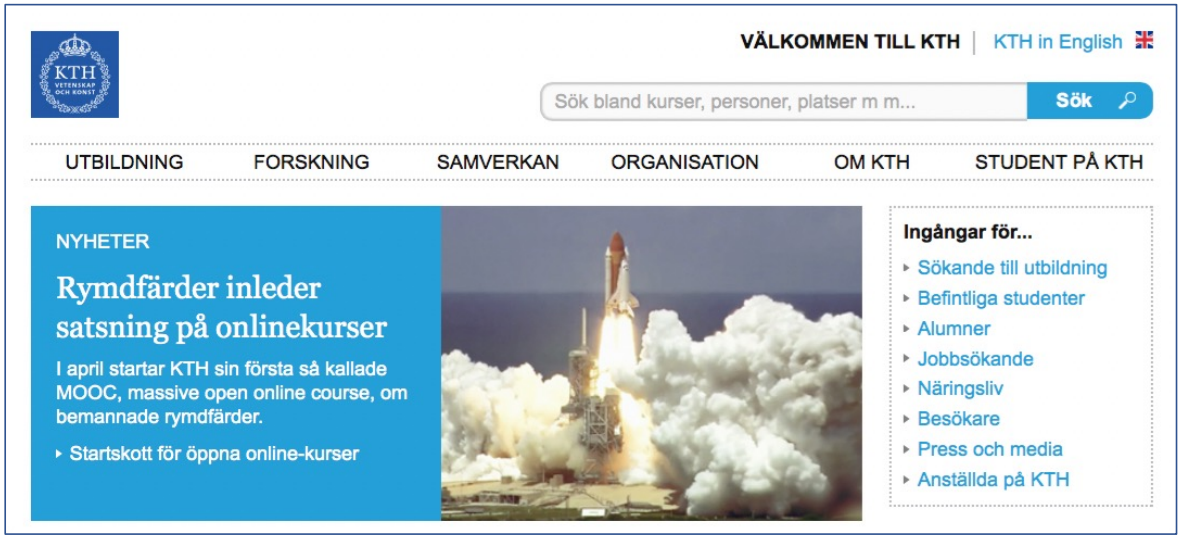



Resursfiler

javascript

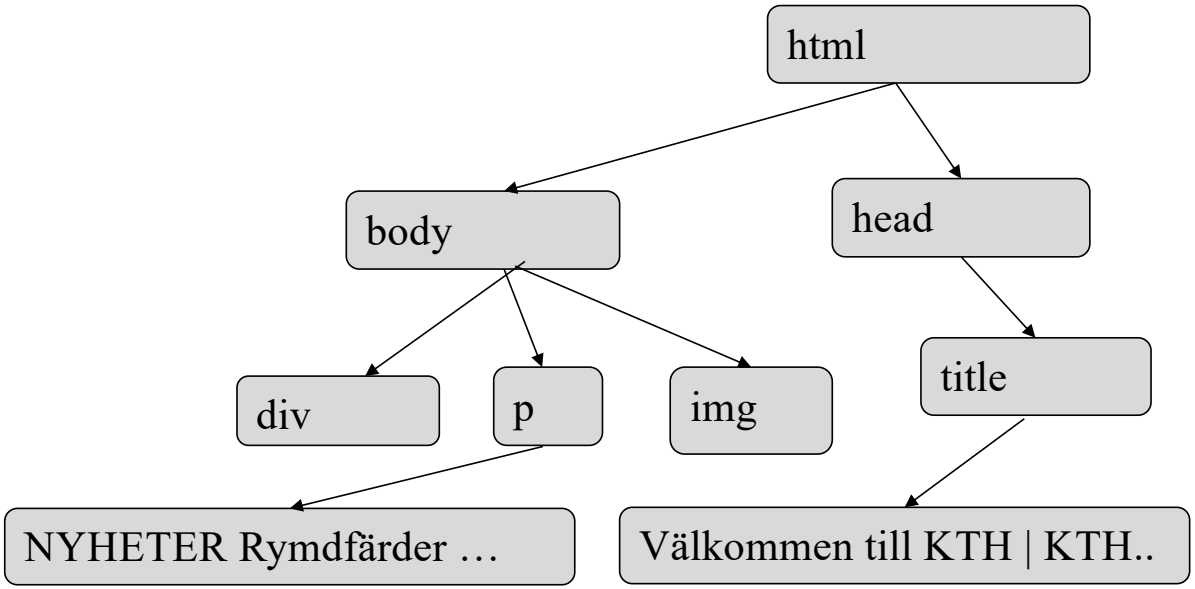
CSS-fil

Renderad sida



# Dom

- Ett trädstruktur baserat på HTML-dokumentet som javascript-koden kan modifiera dokumentet genom att manipulera trädstrukturen.
- Lägga till nod
- Ta bort nod
- Modifiera innehåll
- Visa/dölja



# Javascript

- Till skillnad från java är javascript svagt typat
- Det är mycket vanligt att anropa funktioner i javascript med hjälp av s k event-handlers, dessa registerar inmatning, klickningar och pekarförflyttning. Varje event-handlers är associerade med en typ av händelse och en eller flera taggar. Med detta påverkas den sekventiella exekveringen och händelser kan köras mellan sekvenser av kod.

- m h a javascript kommer man åt taggarna i HTML-dokumentet m.h.a DOM genom att skriva document.getElementById("id")

# Event handler (urval)

| **Event handler** | **Associerade taggar** |
|---|---|
| OnBlur | formulärelement |
| onChange | formulärelement |
| onClick | formulärelement, länkar |
| onFocus | formulärelement |
| onSelect | text, textarea |
| onSubmit | form |

# Exempel

```
<script type="text/javascript">
function check_adress(form){
      if(form.email.value.length==0 || form.email.value.indexOf('@')==-1){
      alert("Du måste ange en giltig epostadress!");
      form.email.focus();
      return false;
      }
      if(form.phone.value.length==0){
      alert("Du måste ange ett telefonnummer!");
      form.phone.focus();
      return false;
      }
      return true;
}
</script>
```

# CSS

- Med stylesheets kan man flytta ut formateringsinformation till en separat fil, på så sätt blir html-koden mer lättläst. Det handlar främst om typsnitt, placering, padding, färgval etc. I headern på html-filen anger man vilken .css fil man vill använda:

- <LINK REL="stylesheet" TYPE="text/css" HREF="stylesheet_fil.css">

- I css-filen kan man konfigurera varje tags karaktär för sig, ett format(typsnitt, färg, padding etc) för <p>-taggen, en annan för <td>-taggen osv. Ofta vill man inte bara ha ett format för en viss tag utan kanske två varianter av <td>-taggen, då får man använda klass-attributet, <td class=”meny”> och <td class=”prislista”>

- Man kan också använda stylesheet direkt i html-koden med style-attributet, <td style=’text-align:center; font-weight: bold;’>, i ”vanlig” html skulle ovanstående få skrivas <td align=”center”><b>

# Stlyle.css

```
td{
padding-top: 3px;
padding-left: 5px;
background-color: "#eeeeee";
font-family: "sans-serif";
font-size: 8pt;
height: 20px;
border-top: "1px #ffffff solid"; }
```

```
th{
background-color: "#ffffff";
font-family: "sans-serif";
font-size: 9pt;
font-weight: bold;
height: 22px;
vertical-align: middle;
text-align: center;
padding: 3px;
border-top: "1px #aaaaaa
dotted";
border-bottom: "1px #aaaaaa
dotted";
}
```

# HTML

```
 <!DOCTYPE html><html>  <head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<table>
<th colspan="2">Post price </th>
<tr><td>Weight Kg</td> <td>Price $/Kg</td></tr>
<tr><td> 1 </td><td> 100      </td></tr>
 <tr><td> 2-5 </td><td> 90 </td></tr>
 <tr><td>6-10</td><td> 80 </td></tr>
<tr><td>10 </td><td> 60     </td></tr>
</table>
 </body></html>
```

# CSS

```
p {
color: red;
text-align: center;
}
```

- Selector: `p`
- Declaration: `color: red;`
  - Property: `color`
  - Value: `red`
- Declaration: `text-align: center;`
  - Property: `text-align`
  - Value: `center`

# Några CSS exempel

Font: typsnitt
p {font-family: "Times New Roman", Times, serif;}

background-color: bakgrundsfärg
body{ background-color: lightblue;}

Padding: avstånd mellan text till avgränsningen
p{ padding: 12px;}
Marginal: avståd mellan avgränsningen till andra elementer
avgränsning: span{ margin: 12px;}
https://www.w3schools.com/css/tryit.asp?filename=trycss_grid_layout_named

# Variabler

Variabler kan deklareras med antingen let, var eller const
let age = 20;
var age = 20;

var: variabeln är tillgänglig inom det skop variabeln deklarerats.
let: variabeln är tillgänglig inom det skop variabeln deklarerats.

# Konstanter

- Konstanter deklareras med reserverad ordet const.
- const PI = 3.141592653589793;
- Följande ger fel:
  - PI = 3.14;
  - PI = PI + 10;

# Automatisk typkonvertering, == och ===

| Uttryck | Resultat |
|---|---|
| 5 * null | 0 |
| ”5” - 3 | 2 |
| ”5” + 3 | ”53” |
| ”5” *2 | NAN (Not An Number) |
| false == 0 | True |
| false === 0 | False |
| ”4711” == new String(”4711”); | true |
| ”4711” === new String(”4711”); | false |

# if-satser, och &&, eller ||

```
let age = prompt(”Age?”);

if (age < 21 || age>=80)
        alert(”coke or juice?”);
else if (age>20 && age<80)
        alert(”coke, juice, or alcoholic drinks?”);
```

# loop

- for (let i=0; i<10; i++){…}
- while(i<10) {…}
- do{….} while(i<10);
- break

# Deklarera funktioner

Tre olika sätt att deklarera functioner:

1. const calc = function(a, b){…}
2. function calc(a, b){…}
3. const calc = (a, b) => {…}

Returvärde

Parametrar

Optional parameters

# Optional parameter

```
function dubblera(x) { return 2*x; }
dubblera(3, 54, ”blahonga")); ger 6
function halvera(a, b) {
   if (b === undefined)
        return a/2;
    else
        return a / b;
}
halvera(22);   ger 11  medan halvera(10, 5);   ger 2
```

# closure

```
function coverValue(n) {
    let local = n;
    return () => local;
}

let cover1 = coverValue(8);
let cover2 = coverValue(5);

cover1() ger 8
cover2() ger 5
```

```
 function divide (divider) {
    return number => number / divider;
}

let half = divide(2);
half(16)   ger  8
```

# Högre ordningens funktion

Funktioner som har andra funktioner som indata eller utdata. Alltså funktioner som tar andra funktioner som argument (indata) eller returnerar andra funktioner (utdata).

```
function h_o_f_1(aFunction){
    aFunction(10 , 12);
}


function h_o_f_2(){
    return (a , b) => {console.log(a+b);}
}
```

# Array

```
let namnlista = [”Anna”,”Kalle”,”Malin”];
```

`namnlista.push(”Johan”)` lägger till ”Johan” i slutet

`namnlista.unshift(”Johan”)` lägger tillsträngen ”Johan” i början

`namnlista.pop()` tar bort och returnerar sista elementet

`namnlista.shift()` tar bort och returnera första elementet

`namnlista.length` ger antal element i listan

`namnlista[2]=”Emma”` tredje element i listan ersätts med ”Emma”

namnlista[100] = ”Peller” ”pelle” placeras i plats nummer 100

# Object

```
let person = {
    name : ”Peter”,
    labbs : [ 1, 2],
}
```

Object_keys(person)  ger [name, labbs]
Object.assign(person, {labbs:[1,2,3,4] , betyg : ”A”})
console.log(person)  ger {name: ”peter”, labbs:[1,2,3,4], betyg: ”A”}

# JSON (JavaScript Object Notation)

- Ett populär serialiserat format.
- Används för att lagra data och kommunion på nätet.
- Formatet är väldigt likt objekt och arrayer i javascript.
- Alla property måste anges med citattecken (”)
- Ingen funktionsanrop, variabel deklaration eller kod som involverar beräkningar.
- Kommentarer är inte tillåtna.
- Exempel:

```
{ ”morgonpigg” : true , ”kurser”:[”tilda21”,
”intnet22”, ”adk22”] }
```