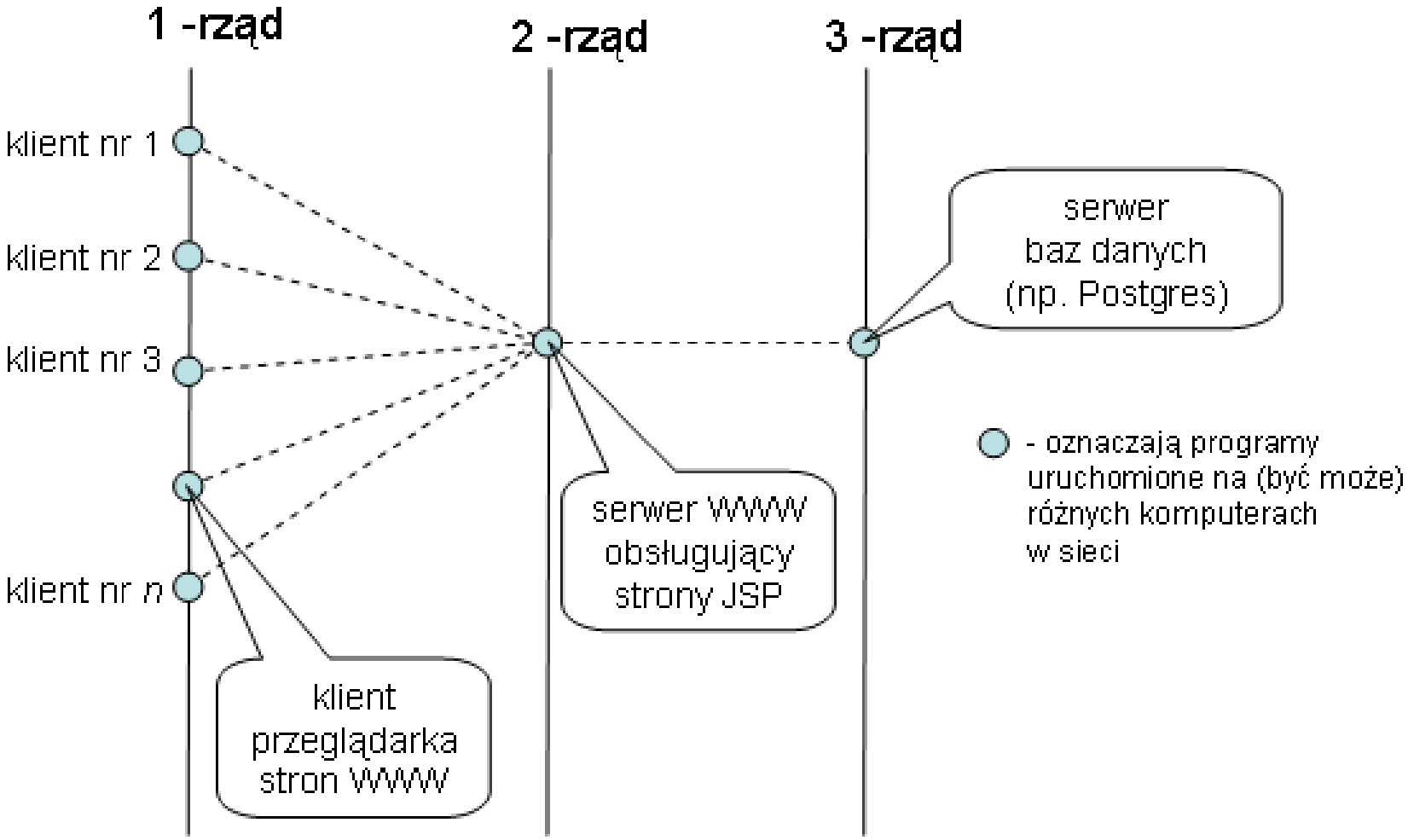
SSI = Sieciowe Systemy Informacyjne

Wykład nr 3

Architektura aplikacji sieciowych



- czym są "rzędy"?
  może chodzić o typy programów (serwer www, baza danych, przeglądarka, ...)
  lub o funkcjonalność (prezentacja danych, logika biznesowa)

- architektura 2 rzędowa:
  1 rząd: "gruby klient", dedykowana aplikacja desktopowa
  2 rząd: baza danych

- architektura 3 rzędowa (aplikacja webowa):
  1 rząd: "cienki klient", przeglądarka www
  2 rząd: serwer www + skrypty
  3 rząd: baza danych

- architektura 4 rzędowa (aplikacja webowa):
  1 rząd: "cienki klient", przeglądarka www
  2 rząd: serwer www + skrypty; funkcja: prezentacja danych
  3 rząd: obiekty rozproszone (Corba, EJB), webserwisy; funkcja: logika biznesowa
  4 rząd: baza danych; funkcja: model danych

- + można bardziej "rozwarstwiać" rzędy...
  + serwer www nie tylko udostępnia statyczne pliki z html-em, ale także tworzy dynamicznie html (np. skrypty PHP, Java/servlety, ...)
  + pamiętać o rozdzielaniu "prezentacji danych" i "logiki biznesowej", nawet jeśli jedno i drugie są w skrypcie PHP!!!
  + logika biznesowa: np. wyciąganie danych z bazy danych przy pomocy j. SQL, prezentacja danych: np. generowanie html-a

Technologie przeglądarkowe (rząd 1, app web).

- HTML - struktura
- CSS - styl (wygląd), *prezentacja = struktura + styl*
- JavaScript - programy w stronach www
- aplety Javowe
- wstawki Flash

HTML - ang. HyperText Markup Language

```
<!DOCTYPE html>
<html>
<head>
  <title>zajęcia</title>
</head>
<body>
<h1>My First Heading</h1>
<p style="color:red">My first paragraph.</p>
<p>tekst1 <i>tekst2</i> tekst3</p>
<p>słowo1<br>słowo2 słowo3</p>
<p>??? <a href="url">to jest link</a> ???</p>
<p><img src="url do obrazka" width=400 height=300 alt="brak obrazka?"> .... </p>
<form action="plik.tcl" method="post">
  pole1: <input type="text" name="pole1" value="111"><br>
  pole2: <input type="text" name="pole2">
</form>
</body>
</html>
```

<html> <head> ... </head> <body> ... <body> </html>

tagi, elementy XML, można je zagnieżdżać, np. ... ,

atrybuty tagów/elementów, np. href, style(!), action, method

http://www.w3schools.com/html

HTML

Rodzaje dokumentów html-owych (pierwsza linia dokumentu):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
.....................................

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
.....................................

<!DOCTYPE html>
.....................................
```

HTML 4.01 - najbardziej popularne

XHTML 1.0 - dokumenty HTML są dokumentami XML (rola XML)

HTML 5 - wiele nowych cech, np. tagi <canvas>, <video>, <audio>

HTML

Formularze HTML - wprowadzanie danych do aplikacji webowej

```
<form method="POST" action="plik2a.tcl">
  wprowadź dane dla pole1 <input type="TEXT" name="pole1" value="123 123"><br>
  wprowadź dane dla pole2 <input type="TEXT" name="pole2"><br>
  wprowadź dane dla pole3 <select name="pole3"><br>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </select>
  <input type="SUBMIT" value="wyślij"><br>
</form>
```

atrybut "method" - POST lub GET (żądanie http tego samego typu !)

atrybut "action" - skrypt obsługujący dane z formularza (url)

elem. <input>, <select> - pola formularza różnych typów,

jeśli mają atrybut "name" to zostaną umieszczone w żądaniu http (name vs id !)

guzik "submit" - jego naciśnięcie powoduje, że przeglądarka wysyła żadąnie http,

z polami formularza i ich wartościami ...

odpowiedzi http mogą być następujące: 200 (nowy html), (302) redirect, ???

HTML

Tabele HTML

```
<table id="tab1">
  <tr>
    <th>Miesiąc</th> <th>Kwota</th>
  </tr>
  <tr id="row1">
    <td>Styczeń</td> <td>100zł</td>
  </tr>
  <tr id="row2">
    <td>Luty</td> <td>200zł</td>
  </tr>
</table>
```

wygląd tabeli robić przez CSS, nie przez atrybuty !!!

HTML

Inne ważne tagi HTML

```
<p> paragraf </p>
<p> linia 1 <br> linia 2 </p>
<hr> linia pozioma oddzielająca tekst...

<p> to jest lista numerowana:
<ul>
  <li>punkt 1</li>
  <li>punkt 2</li>
</ul></p>
<p> to jest lista NIEnumerowana:
<ol>
  <li>punkt 1</li>
  <li>punkt 2</li>
</ol></p>

<pre>
tekst "preformatowany"
</pre>

<i>italika, tekst pochyły</i>
<i>bold, tekst pogrubiony</i>
```

```
<div id="blok1"> blok tekstu .... </div>
<span id="s1"> blok tekstu .... </span>

  &lt; &gt; &amp;
oraz inne encje/entities
patrz: http://www.w3schools.com/html/html_entities.asp

<a href="http://host:port/sciezka/plik.htm#zakładka>to jest link do zakładki</a>
<a href="http://host:port/sciezka/skrypt.php?par1=111&par2=222">link do ...</a>
  // zakładka też może być, przed "?"

<form action="http://host:port/sciezka/skrypt.php">
...
</form>

http://www.w3schools.com/tags/
```

CSS - ang. Cascading Style Sheets

Rezultat:


```
<p> tekst1 <span id="t2">tekst2</span> tekst3 </p>
<p> tekst1 <span class="ccc">tekst2</span> tekst3 </p>
<p> tekst1 tekst2 <span class="ccc">tekst3</span> </p>
<p class="ddd"> tekst1 tekst2 tekst3</p>
<style  type="text/css">
#t2 {color:red;}
.ccc {color:blue;}
p.ddd {color:green;}
p {color:yellow}
</style>
```

w elemencie <style> lub w zewnętrznym pliku umieszczamy reguły:

selektor { właściwość:wartość; właściwość:wartość; ... }

selektor to: #id, .class, tag.class, tag1 tag2 (zagnieżdżenie), tag1 "," tag2, ...

właściwości: color, background-color, margin, position, ... b. dużo ...

rozmieszczanie składn. strony: dawniej <table>, dzisiaj:

składnik strony + CSS !!!

css w zewn. pliku: <link rel="stylesheet" href="plik.css" type="text/css">

http://www.w3schools.com/css

CSS

Obrazek opływany przez tekst zrobiony przy pomocy CSS

```
<p>
<img src="img.jpg" width=120px height=100px alt="???" id="img1">
Tekst tekst tekst tekst tekst tekst tekst tekst tekst tekst tekst tekst tekst
.................................................
</p>

<style>
  #img1 {
    float: left; margin=3px 13px 3px 3px;
    border-color: green; border-width: 15px; border-style: groove;
  }
</style>
```

float: tekst będzie opływał obrazek (obrazek będzie po lewej)

margin: rozmiary marginesu (top right bottom left)

border-*: obramowanie obrazka (niezależnie od marginesu)

CSS

Stylizowanie tabeli przy pomocy CSS

```
<table id="tab1">
  <tr>
    <th>Miesiąc</th> <th>Kwota</th>
  </tr>
  <tr id="row1">
    <td>Styczeń</td> <td>100zł</td>
  </tr>
  <tr id="row2">
    <td>Luty</td> <td>200zł</td>
  </tr>
</table>
<style>
  table, th {border:5px groove pink;}
  #row1 {color: red;}
  #row2 {color: white; background-color: blue;}
</style>
```

border: ustawia obramowanie całej tabeli oraz jej elementów (grubość typ kolor)

border można ustawiać dla tagów: table, td, th (ale nie dla tr !!)

zmienia sie kolor i kolor tła w wierszach row1 i row2

CSS

Bloki tekstu i ich umieszczanie na stronie przy pomocy CSS

```
<p> paragraf 1</p>
<p id="par2"> paragraf 2 tekst tekst tekst tekst tekst tekst tekst tekst tekst</p>
<p> paragraf 3</p>
<style>
  #par2 {
    color: red;
    position: relative; top: 0px; left: 50px; width 5cm
   }
</style>
```

blok tekstu może być w elemencie <p>, ale może być też w

position=relative oznacza, że przemieszczamy bok wzg. jego naturalnej pozycji

(jeśli top=0px, left=0px to blok będzie w swojej naturalnej pozycji w stronie...)

top i left to przesunięcie pozycji bloku, może być <0, jednostki: px, cm, ...

width: szerokość bloku, jeśli tekst jest długi to się "złamie" ....

strona www może się składać z samych boków w , z ustawionym position=absolute,

i odpowiednio ustawioną pozycją ...

JavaScript

- język programowania - język z obiektami (ale nie OOP)

```
function fun1(a,b) {return a+b;}; fun1(1,2); // def i wywoływanie funkcji
var zm_lok=123; zm_glob=123; // zm. lokalne/globalne w funkcji
x=123; if(x>100) {y=1;} else {y=0;}
o1={}; o1.a= 111; o1.b="qqq www"; o1.c=[1,2,3]; o1.d={x:11, y:22};
```

- biblioteki JS: JQuery, YUI, OpenLayer (GIS), ...
- obsługa zdarzeń w stronie www

```
<button onClick="kod JS">guzik nr 1</button>
```

- operacje na drzewie DOM dokumentu HTML (dokumentu XML)

```
d1 = document.getElementById("js_test1_d1").childNodes;
d1.length; // zwraca liczbę elementów
d1[1].id; // zwraca id elementu (np. ustawiony ręcznie w html-u)
d1[1].nodeType; // zwraca typ obiektu: 1- element, 2 - atrybut, 3 - tekst, ...
```

- można modyfikować/odczytywać CSSy elementów:

```
document.getElementById("p1").style.position="relative";
document.getElementById("p1").style.left="50px";
```

JavaScript

Typy danych w JS

- tablice
  t1= [1,2,3,"qqq www eee", [1,2,3]]; t2= new Array(1,2,3)
  odwołania do elem tablicy: t[1], t[4][1]
  operacje na tablicy: patrz obiekt predefiniowany Array,
  np. metoda "push", prop "length"

- obiekty
  o1= {}; o1.a=123; o2= {x:"qqq", y:{q1: 11, q2: 22}, z:[1,2,3]};
  o1.met1= function(b_, c_) {this.b= b_; this.c=c_}
  tak definiujemy metody obiektu, "this" dotyczy bieżącego obiektu,
  funkcja anonimowa, zagnieżdżanie obiektów/tablic

- stringi
  s1= "abcedef"; s1.slice(2,4) = "ce"; s2=new String('qqq')
  operacje na stringach: patrz obiekt predefiniowany String,
  np. metoda "slice", prop "length"

- oraz inne predefiniowane obiekty....
  patrz: http://www.w3schools.com/jsref/
  pokazać przykład js_test...

JavaScript

Obiekty w JS

1. konstruktor - funkcja, którą podaje się po "new", zmienna "this"
2. metoda obiektu - właściwość, której przypisano funkcję (np. anonimową)
3. prototype - właściwość konstruktora (dosłownie!)
   zasada działania: jeśli danej właściwości nie ma w obiekcie,
   to się jej szuka w prototypie ...

```
o1={a:1111, b:2222};
function MojObiekt(c_, d_) { this.c=c_; this.d=d_;}
MojObiekt.prototype= o1;
o2= new MojObiekt(333, 444);
// object o2 ma właściwości: {a number} {b number} {c number} {d number}
o3= new MojObiekt(3333, 4444);
o1.met1=function() {return [this.a, this.b, this.c, this.d]}
o2.met1(); // co sie uruchomi?
o3.met1=function() {return [this.a, this.b, this.c, this.d, "!!!"]}
o3.met1(); // co sie uruchomi?
delete o3.met1;
o3.met1(); // co sie uruchomi?
```

JavaScript

Dostęp do stylu (CSS) z pozmiomu JS

```
document.getElementById("p1").style.color="red";
document.getElementById("img1").style.margin="3px 13px 3px 3px";

document.getElementById("img1").style.verticalAlign="top";
  // uwaga: css "vertical-align", js "verticalAlign" (prawie zawsze tak jest!)

document.getElementById("img1").style.cssFloat="right";
  // tekst oplywa obrazek, obrazek po lewej
  // UWAGA: css "float", js "cssFloat" !!!
```

JavaScript

Dostęp do drzewa DOM dokumentu HTML

```
d0 = document.getElementById("p1");
d1 = d0.childNodes; // d1 to tablica dziećmi elem "p1"
d1.length; // 5

x1=[];  for(i=0; i<d1.length; i++) x1.push(d1[i].nodeType);
x1; // 3,1,3,1,3
  // "1" oznacza ze child jest typu "element", "3" ze typu "text"

d1[0].id; // null
d1[0].nodeType; // 3
d1[0].nodeValue; // wartość, czyli tekst

d0.appendChild(document.createTextNode("qqq"));
d0.removeChild(d1[3]);
  // modyfikowanie drzewa DOM...

document.getElementById('p2').innerHTML='nowy html'
  // modyfikowanie html wewnątrz elem.
```

JavaScript

Biblioteki JS: JQuery

zasada działania wyrażeń JQuery: $(selektor elementów).metoda(parametry);

```
$('#p2').css('color','green'); // zmiana stylu CSS elem.
$("#p2").html("A ku ku !!!"); // zmiana html-u wewnątrz elem.

$("p:even").css('color','pink'); // dotyczy elem <p> o parzystych nr
$("p span:odd").css('color','red');
   // dotyczy elem <span> wewnątrz <p>, o nieparzystych nr

$("#g1").unbind('click');
$("#g1").click(function(){  /*fun*/ });
   // definiowanie obsługi zdarzenia dla elem z id="g1"
   // uwaga: można to robić "daleko" od samego elem...
```

JavaScript

AJAX (ang. Asynchronous JavaScript and XML.)

technika wysyłania żądań http bez przeładowywania strony www (klikania itp)
JQuery bardzo ułatwia używanie ajax-a ...

```
<p id="p1">tekst</p>
<p><button id="guzik1">guzik1</button> <button id="guzik2">guzik2</button></p>

<script>
// zaladowanie do elem "p1" odpowiedzi ajaxowej (html-a) ...
$("#guzik1").click(function() {
     $("#p1").load("/ajax/a1", {x1:11, x2:2222});
        // można też przekazać parametry do skryptu a1 !
})
$("#guzik1").click(function() {
     $("#p1").load("/ajax/a2");
})
// wykonanie skryptu JS, który przyjdzie w ajaxowej odpowiedzi ...
$("#guzik1").click(function() {
     $.ajax({url:'/ajax/b', dataType:'script', data:{id:'p1', color:'red'}});
})
$("#guzik2").click(function() {
     $.ajax({url:'/ajax/b', dataType:'script', data:{id:'p1', color:'blue'}});
})
</script>
```

JavaScript

"Walidacja" danych wpisywanych do formularza

```
<form id="form1" action="url">
label1 <input type="text" name="t1" value="123123"><br>
label2 <input type="text" name="t2"><br>
<input type="submit" value="sprawdź" id="g1">
</form>

<script>
$("#g1").click(function(){
  var t1= this.form.t1.value; // tak sie odwołujemy do pola w form !
    // to jest "stara" metoda odwoływania się do pól...
  document.flash_kli.msg('t1='+t1); // komunikat diagnostyczny
});
</script>

<script>
$("#form1").submit(function(){
  var t1= this.t1.value; // !!!
  return false; // jeśli dane w form NIE ok...
  //return true; // jeśli dane w form ok, to wyślij żądanie http
});
</script>
```

JavaScript

"Walidacja" danych wpisywanych do formularza (przez drzewo DOM)

```
<form id="form1" action="url">
label1 <input type="text" name="t1" value="123123" id="tt1"><br>
label2 <input type="text" name="t2" id="tt2"><br>
<input type="submit" value="sprawdź" id="g1">
</form>

<script>
$("#g1").click(function(){
  var t1= document.getElementById('tt1').value;
    // inaczej odwołujemy się do elem !!!
  document.flash_kli.msg('t1='+t1);
});
</script>

<script>
$("#form1").submit(function(){
  var t1= document.getElementById('tt1').value;
  return false; // jeśli dane w form NIE ok...
  //return true; // jeśli dane w form ok, to wyślij żądanie http
});
</script>
```