JavaScript

Level 2

JavaScript OOP

A JavaScript **nagyon** Objektum Orientált nyelv.

```
var myFirstBoolean = true;
var mySecondBoolean=new Boolean();
```

A leg elemibbnek tűnő típusok is objektumok,
tulajdonságokkal, metódusokkal.

OOP Osztályok nélkül

Mi az, hogy osztály?

Az objektumaink tervrajza.

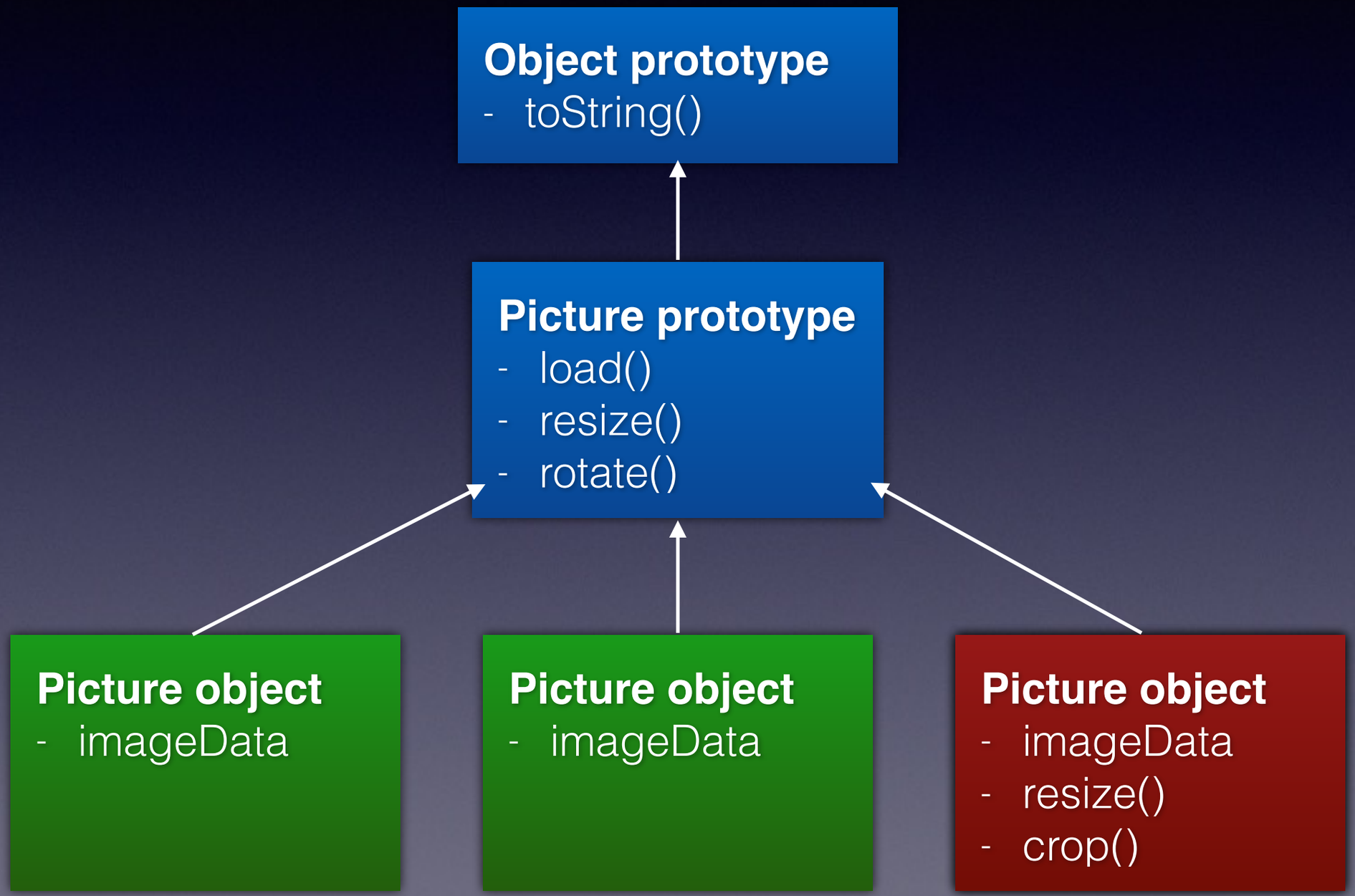
Mi az, hogy prototípus?

Egy olyan objektum, amihez hasonlótból többet szeretnénk csinálni

Prototype

- Egy árnyékobjektum, aki tud mindent, amit az objektumunk nem tud
- Minden objektumnak van prototípusa
- Még a prototípusoknak is van, prototípus láncok alakíthatóak ki (öröklődés)
- A prototípus lánc végén mindig az Object áll

Prototype


Object prototype
- toString()
Picture prototype
- load()
- resize()
- rotate()
Picture object
- imageData
Picture object
- imageData
Picture object
- imageData
- resize()
- crop()

Prototype

Így néz ki JavaScriptben egy “osztály”

```
// konstruktor
function Picture(src){
   this.src = src;
   if(typeof src == ‘string’) this.load();
}

// prototípus
Picture.prototype = {
   load: function(){
      this.imageData = loadImageSomeHow();
   },
   resize: function(scale){…},
   rotate: function(rad){…}
}
```

Prototype

Munka az objektum példányokkal

```
// Példányosítás
var myPic = new Picture(src);

// Módosítás
myPic.crop = function(frame){…};
```

```
// Hívás a prototype prototípusába
alert(myPic);

// Újabb módosítás
myPic.toString = function(){
   return ‘Picture ’+this.src;
};
alert(myPic);
```

Prototype chaining

Prototípus láncolás - öröklés

```
function Sprite(src){
   Picture.call(this, src);
   this.x = 0;
   this.y = 0;
   thix.alpha = 1;
}

Sprite.prototype = Object.create(Picture.prototype);

Sprite.prototype.move = function(x, y){…};
Sprite.prototype.fade = function(alpha){…};
```

ECMAScript 6
class

```
class Point{
    constructor(x, y){
        this.x = x;
        this.y = y;
    }
    toString(){
        return ‘(‘+this.x+’:’+this.y+’)’;
    }
}
```

ECMAScript 6
öröklődés

```
class RainbowPoint extends Point{
    constructor(x, y, color){
        super(x, y);
        this.color = color;
    }
    toString(){
        return ‘(‘+this.x+’:’+this.y+’ ’+this.color)’;
    }
}
```

ECMAScript 6
getter / setter

```
class PRainbowPoint extends RainbowPoint{
    get color(){
        return this.color;
    }
    set color (color){
        if(this.isValidColor(color)) this.color = color;
    }
    isValidColor(color){
        return true;
    }
}
```

Esemény vezérelt programozás

Reagálunk a környezet változásaira

azaz az eseményekre!

Felhasználó által kiváltott események
(egér, billentyű, stb…)

A böngésző által küldött események
(ablak betöltődése)

Az általunk írt programok által kiváltott események

Egy HTML dokumentum eseményei a teljesség igénye nélkül

- Ha a user kattint az egérrel
- Ha a lap betöltött
- Ha egy kép betöltött
- Ha az cursor egy elem fölé ér
- Ha egy input mező megváltozott
- Ha egy HTML űrlapot elküldenek
- Ha a felhasználó leüt egy billentyűt

Eseménykezelő

Az objektum, akin az eseményt kezelni szeretnénk

Az esemény, amire feliratkozunk

Az eseményt leíró objektum

```
a.onclick = function (event) {
    alert(this.id + '-re kattintottál');
};
```

Az objetum, akin az esemény történt

```
function cklickHandler(event) {
   alert(this.id + '-re kattintottál');
};
a1.onclick = clickHandler;
a2.onclick = clickHandler;
```

Esemény vezérelt JS

- Felhasználói interakció

```
a.onclick = function () {
    alert('Rám kattintottál');
};
```

- A DOM-ban történt bizonyos változások

```
window.onload = function () {
    alert('Kész vagyok');
};
```

Eseménykezelő

```
var a = document.createElement('a');
a.href = 'http://...';
a.target = '_blank';
a.title = 'Az én linkem';
a.onclick = function (event) {
    event.preventDefault();
    alert('Marad, ahol van.');
};
var bodies =
    document.getElementsByTagName('body');
bodies[0].appendChild(a);
```

Esemény objektum

- Tartalmazza az eseményt jellemző információakt
- Tartalmazza az esemény feldolgozását jellemző információkat
- Tartalmazza az esemény kezeléséhez szükséges metódusokat

Event bubbling

Böngésző esemény kezelés

Saját esemény kezelés

```
<body>
    <a href="…">
        <img src="…" title="…" />
    </a>
</body>
```

Böngésző esemény kezelés

Saját esemény kezelés

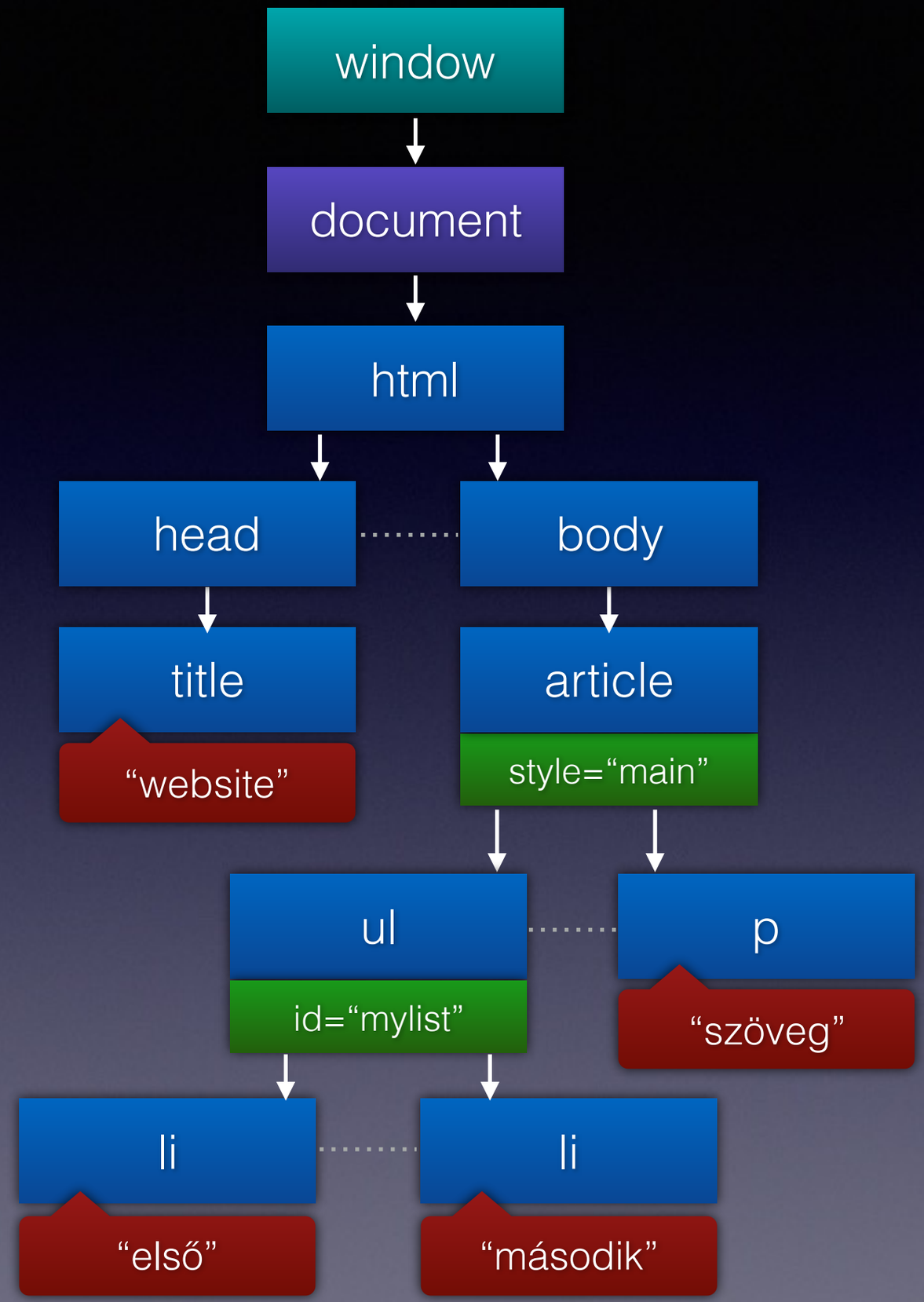
Böngésző esemény kezelés

Saját esemény kezelés

```
event.preventDefault()
event.stopPropagation()
```

DOM

```
<!DOCTYPE html>
<html>
    <head>
        <title>website</title>
    </head>
    <body>
        <article style="main">
            <ul id="mylist">
                <li>első</li>
                <li>második</li>
            </ul>
            <p>szöveg</p>
        </article>
    </body>
</html>
```



window.onload

```
var a = document.createElement(‘a’);
…
var bodies =
document.getElementsByTagName(‘body’);
bodies[0].appendChild(a);

window.onload = function () {
    // DOM-mal dolgozó alkalmazáslogika
    // indítása/inicializálása
}
```

Adatok a kliensen

Cookies - sütik

A szerver és a kliens is írja/olvassa. A kliens tárolja.

A http headerben utazik, oda - vissza.

Nehézkes a kezelése, session kezeléshez kiváló.

Adatok a kliensen
localStorage és **sessionStorage**

String kulcs

String érték

localStorage.setItem(key, value);
localStorage.getItem(key);

```
localStorage.setItem('userName', 'Elvis Presley');
var userName = localStorage.getItem('userName');
```

Adatok a kliensen: **localStorage**

```
var user = {}
user.firstName = 'Elvis';
user.lastName = 'Presley';
var userJSON = JSON.stringify(user);
localStorage.setItem('user',userJSON);


var userJSON = localStorage.getItem('user');
try{
   user = JSON.parse(userJSON);
   // Ha a userJSON változóban NEM értelmezhető JSON
   // string van, akkor a JSON.parse kivételt dob!
} catch(e){
   user = null;
}
```

Adatok a kliensen: sql

http://www.tutorialspoint.com/html5/html5_web_sql.htm