JavaScript & DOM

XML & JSON

Ele alınacak başlıklar

- JavaScript Dili
- DOM (Document Object Model)
- JavaScript ve OO Programlama
- XML ve JSON

JavaScript

Kısım 1

Gelişim

- JavaScript, 1995 yılında Netscape tarafından geliştirilmiştir.
- JavaScript
  - Dinamik (dynamic)
  - Tip güvensizdir. (weakly type-checked)
  - Fonksiyoneldir. (functional)
  - Nesne yönelimlidir.(object-oriented)
  - Karakter duyarlıdır. (case-sensitive)
  - C tabanlı bir yazım şekli vardır.

```
eval("1 + 2");
i = "23" * 2 + false;
map( f, [1, 2, 3, 4, 5] );
s = obj.toString( );
convention.useCamelCasing( );
if (condition) {
  statement;
  statement;
}
```

Türler

- JavaScript 4 tane veri türü sunmaktadır:
  - boolean : *true / false*
  - number: *64 bit reel sayılar (NaN ve Infinity dahil olmak üzere)*
  - string : *0 veya daha fazla Unicode karakteri*
  - object :

```
x = 1.0;
document.write(typeof(x)); // prints 'number'
```

Özel değerler

- null : null referans (no object)
- undefined : değişkenin değeri yoktur
- false, 0, "", null, undefined olan değişkenleri false olarak işlem görür.

Değişken Tanımlama

- JavaScript'de değişken tanımlama seçimliktir.
  - Böyle bir durumda, yanlış yazılan değişken isimleri yeni bir değişken tanımlamakla sonuçlanır.
  - Kullanılacak değişkenlerin önceden tanımlanması tavsiye edilir.
- Tanımlama
  - "var" anahtar sözcüğü kullanılır.
  - Tipi belirtilmeyen değişkenlerin ilk değerleri "undefined" olur.
  - Local ve global olmak üzere 2 scope söz konusudur.

```
var someVar = 1.0;

someVar = someVar + 1;
someVar = "And the answer is " + someVar;
alert(someVar);
```

Statements

```
var sum = 0.0;
var someArray = [1, 2, 3, 4, 5];

for (var i=0; i < someArray.length; i++)
  sum += someArray[i];

alert(sum);

// we also have a foreach-like statement:
var sum2 = 0.0;

for (var i in someArray)  // for each index, not element:
  sum2 += someArray[i];

alert(sum2);
```

Operatörler

- +, -, *, /, vb.

```
alert(1 + "23" + 4);  // displays "1234"
```

- &&, ||

```
var name  = obj && obj.name;  // guard against null ref
var value = i || 1;           // default to 1 if i undefined
```

- ===, !==

```
var s = "";
alert( s == 0  );  // true!
alert( s === 0 );  // false, as you would expect
```

Diziler

- Diziler, aynı zamanda dictionary'dir. Dictionary'lerden farkı length özelliklerinin olmasıdır.
- String tabanlı numeric değerlerle indeksleme yapılabilir.
- Dizilerde, kullanılmayan boşluklar olabilir.

```
var someArray = [10, 20, 30, 40, 50];

alert( someArray["3"] );  // 40

for (var i in someArray)  // indices are strings!
  alert( typeof i );
```

```
var A1 = new Array(), A2 = [];  // create an empty array:

for (var i = -100; i <= 100; i++)  // add 201 elements at -100…100:
  A1[i] = i;

alert(A1.length + ", " + A1[-100] + "..." + A1[100]);
```

101, -100...100

```
var A2 = []; // create another empty array:

A2[123] = "Pooja";  // sparse array with 3 elements, length of 790:
A2[456] = "Kim";
A2[789] = "Drago";
```

Fonksiyonlar

- JavaScript'de fonksiyonlar türsüzdür.
  - Fonksiyon her türden eleman döndürebileceği gibi hiç bir şey de döndürebilir.

```
//
// returns sum of all elements in the given array:
//
function sumAll(someArray)
{
  var sum = 0.0;

  for (var i in someArray)
    sum += someArray[i];

  return sum;
}
```

Fonksiyonlar nesne olarak işlem görürler. Fonksiyonların parametreleri de fonksiyon olabilir.

```
// apply the function F across all the elements of collection C:
function applyToAll(F, C)
{
  for (var i in C)
    F( C[i] );
}
```

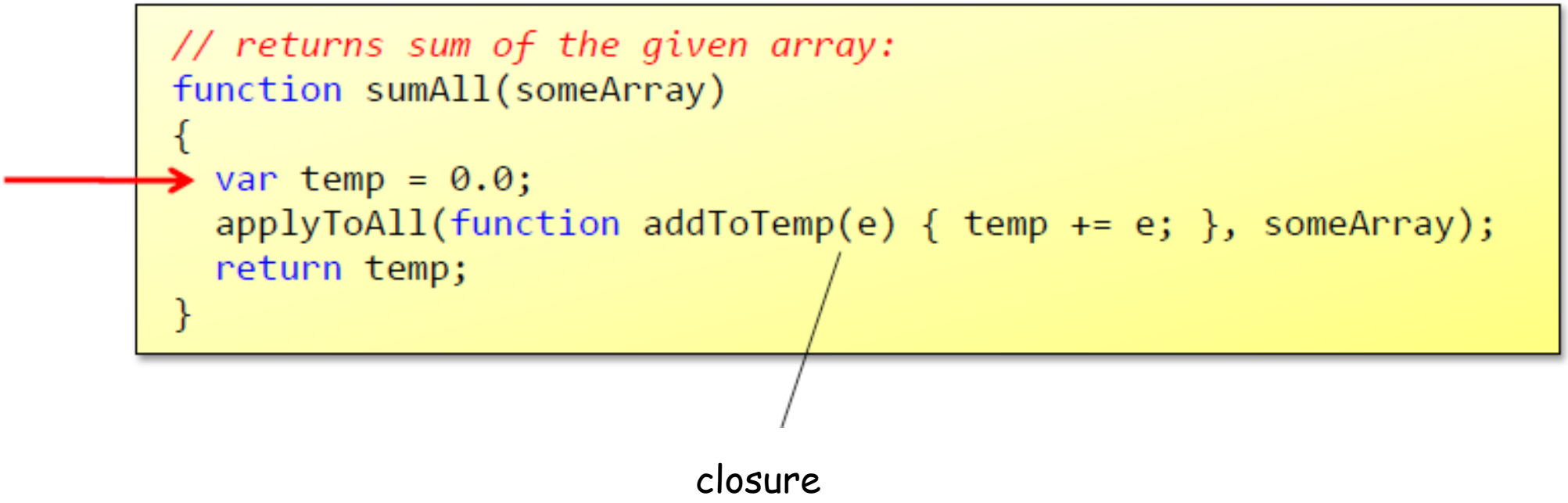
```
// alert each element of the given array:
function alertAll(someArray)
{ applyToAll(alert, someArray); }

// another way to write the above, demonstrating JavaScript's flexibility:
function alertAll_v2(someArray)
{ applyToAll(new Function("e", "alert(e);"), someArray); }

// returns sum of all elements in the given array:
function sumAll(someArray)
{
  var temp = 0.0;
  applyToAll(function addToTemp(e) { temp += e; }, someArray);
  return temp;
}
```

Closures

Fonksiyonlar çağrıldıklarında icra edilirken, deklere edildiklerinde noktadaki değişkenlere erişmesi mümkündür.



```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="style.css">
<script src="script.js"></script>
 </head>
<body>
<h1 id="output">Result = </h1>
<script>
var scope = "global";
function testScope() {
     var scope = "local";
     function innerFunc() {
               return scope; }
     return innerFunc; //fonksiyon donmektedir, değer değil
}
     var innerFunc = testScope();
     var answer = innerFunc();
     output.innerHTML += answer;
</script>
 </body>
</html>
```

Bir fonksiyon içinde başka bir fonksiyon oluşturulduğunda bir **closure** oluşturulmuş olur. İçteki fonksiyon (closure olan) dışarıdaki fonksiyonun yerel değişkenlerine ulaşabilmekte ve bu değişkenleri sonradan da hatırlayabilmektedir.

http://plnkr.co/edit/TPPAW6H5TWrnS79ow44i?p=preview

DOM
(Document Object Model)

Kısım 2

DOM (Document Object Model)

- DOM, HTML ve XML dkümanlarını programlamak için oluşturulmuş bir API'dir (http://www.w3.org/TR/WD-DOM/introduction.html).

- Dokümanların mantıksal yapısını ve dokümana nasıl erişileceği ve üzerinde ne gibi değişikliklerin yapılabileceğini tanımlar.

- DOM ile programcılar, doküman yaratabilirler, yapı içerisinde dolaşabilirler, eleman ve içerik ekleyebilir, düzenleyebilir ve silebilirler.

- DOM, web sayfalarının tüm bileşenlerini erişilebilir kılar.
  - HTML elemanları,
  - attribute'lar
  - Text
- Bütün bu işlemler JavaScript ile gerçekleştirilir.

DOM Nesneleri

- DOM bileşenlerine nesneler veya nesne kolleksiyonları üzerinden erişilebilir.
- DOM bileşenleri bir ağaç yapısı oluştururlar.
  - Parent node ve child nodlar arasında ilişki bulunur.
  - document root nodedur.
- Elemanların niteliklerine (attributes) text olarak erişilebilir.
- Tarayıcılar, DOM'u görsel olarak bir ağaç olarak gösterebilir.
  - (IE'de Araçlar>Geliştirici Araçları sekmesi)
  - (Chrome'da Diğer Araçlar > Geliştirici Araçları)

DOM Standartları

- W3C (www.w3.org) standartları belirler.
- Standartlar 1998'den itibaren geliştirilmiş olup günümüzde de halen geliştirme süreci devam etmektedir. (http://www.w3.org/standards/techs/dom#w3c_all)

Fakat web tarayıcısı geliştiricileri,

- Standartların tümünü uygulamazlar.
- Bazı standartları farklı şekilde uygulayabilirler.
- Bazı ek özellikleri kendi tarayıcılarına ekleyebilirler.

Node'lara id ile Erişim

document.getElementById(<id>)

Geriye id ile belirtilmiş elemanı döndürür.

id attribute'ü her pek çok tag içerisinde bulunur (div, span, form vb..)

document.getElementsByTagName(<tag>)

Geriye,<tag> olan tüm elemanları kolleksiyon olarak döndürür.
Kolleksiyondaki elemanlara indeks numarası ile erişilebilir. Örn:

```
var html= document.getElementsByTagName("html")[0];
var divs = document.getElementsByTagName("div");
```

document.getElementsByName(<name>)

Geriye,<name> olan tüm elemanları kolleksiyon olarak döndürür.
(Aynı isimde birden fazla eleman olabilir).
Örn:

```
var forms= document.getElementsByName("form")
var tables = document.getElementsByName("table");
```

Dolaşım
(DOM Traversal)

- childNodes
- firstChild, lastChild
- nextSibling, previousSibling
- parentNode gibi özellikler ile DOM ağacı üzerinde dolaşılabilir.

JavaScript Objelerinin Özelliklerine Erişim

JS objelerinin özelliklerine erişim için kullanılan iki farklı yazım şekli bulunmaktadır.

- <object>.<property> (document.noteType)
- <object>[<property>] (document[“nodeType”]

Elemanların Nitelikleri (Attributes of Elements)

- "attributes" özelliği ile erişilir.
- Name ve Value olmak üzere 2 adet yapı söz konusudur. Örn:

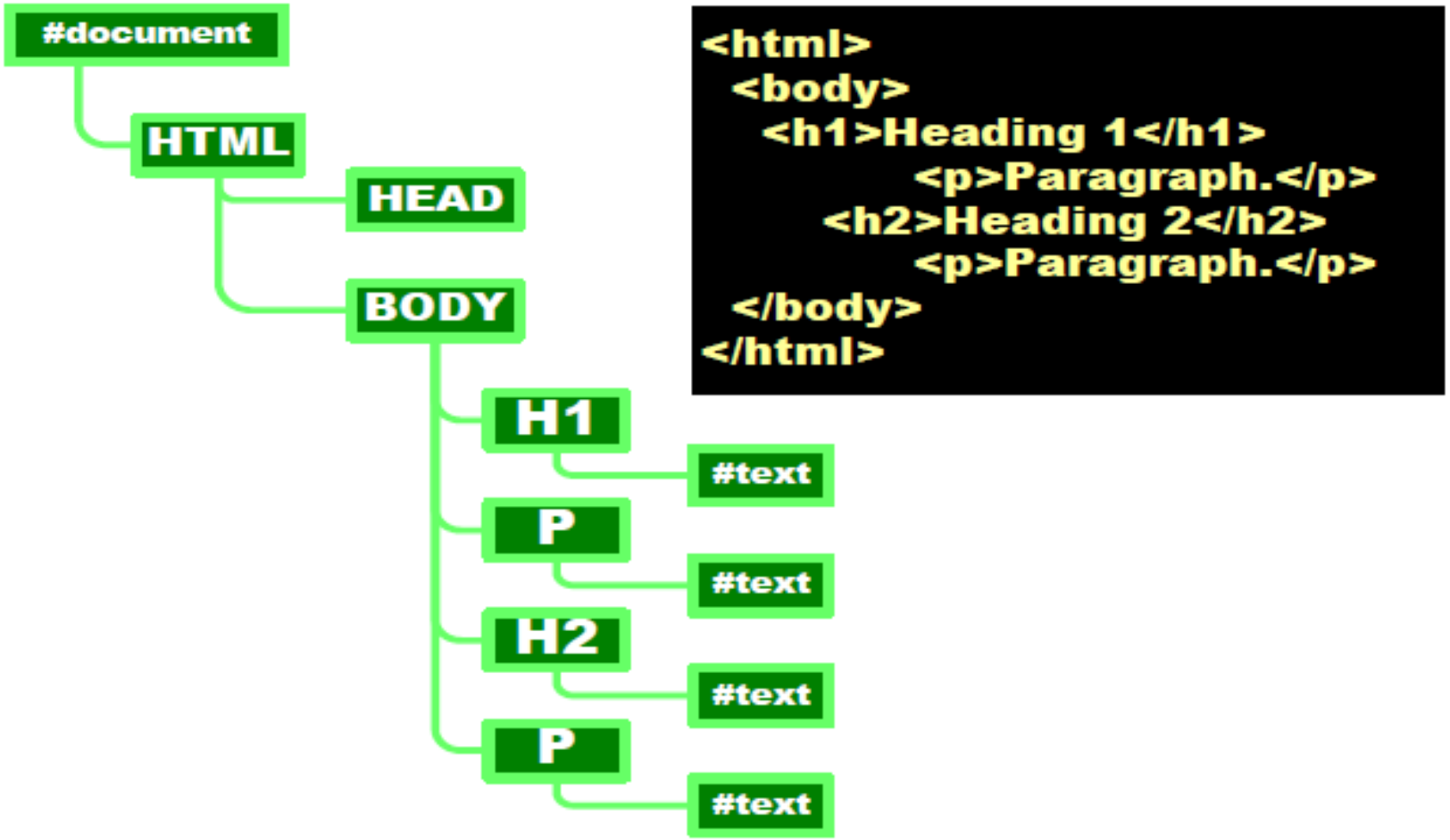
```
var src=document.images[0].attributes[0].value;
var src=document.images[0].getAttribute("src");
```

Text Node'lar

DOM ağacında sadece bir yaprak olarak yer alırlar.

- "nodeValue" özelliği içerdiği metni tutar.
- "innerHtml" ile ise içerdiği metne erişilebilir.

DOM Ağacı


#document
HTML
HEAD
BODY
H1
#text
P
#text
H2
#text
P
#text
<html>
<body>
<h1>Heading 1</h1>
Paragraph.
<h2>Heading 2</h2>
Paragraph.
</body>
</html>

JavaScript OOP

Kısım 3

Nesneler

- JS'de, C#'dan alışkın olduğumuz class gibi bir anahtar kelimeye sahip değildir.

- Nesneler, key/value ikililerinin bir kolleksiyonudur.

  Key'ler field ve metot isimlerini temsil ederler. Value'lar ise field value'lerini ve metot gövdelerini temsil ederler.

```
var employee = new Object();

employee["name"]   = "Pooja";     // define Name key/value pair
employee["salary"] = 72001.44;    // define Salary key/value pair

employee["issuePaycheck"] = function() {  // define issuePaycheck pair
  var monthly = this["salary"] / 12;
  alert(this["name"] + ": " + monthly);
};
```

```
employee["issuePaycheck"]();
```

Nesne Tanımlama Syntaxı

3 farklı şekilde olabilir.

```
var employee = new Object();

employee["name"] = "Pooja";
employee["salary"] = 123.00;
:
:
employee["issuePaycheck"]();
```

```
var employee = {};  // empty obj

employee.name = "Pooja";
employee.salary = 123.00;
:
:
employee.issuePaycheck();
```

```
var employee = {
  "name"   : "Pooja",
  "salary" : 123.00,
  :
  :
};

employee.issuePaycheck();
```

Sınıf Tanımlaması

```
function Employee(name, salary)
{
  this.name = name;          // add Name key/value pair, with value name
  this.salary = salary;      // add Salary key/value pair, with value salary

  this.issuePaycheck = function()   // add issuePaycheck key/value pair…
  {
    var monthly = this.salary / 12;
    alert(this.name + ": " + monthly);
  };
}
```

Nesne Tanımlama

```
function Employee(name, salary)
{
  .
  .
  .
}
```

```
var employee;
employee = {};
Employee.call(employee, "Pooja", 72001.44);
```

Daha bilinen ve alışıldık olan tanımlamaya denktir.

```
var employee;
employee = new Employee("Pooja", 72001.44);
```

Xml ve JSON

Kısım 4

XML
(Extensible Markup Language)

- HTML veriyi göstermek için tasarlanmıştır.
- XML veriyi tanımlamak için tasarlanmıştır. Bu haliyle yazılımdan ve donanımdan bağımsız bir yapısı vardır.
- Kullanıcının kendi xml taglarını tanımlaması mümkündür. Nesneleri kendi taglarımızı kullanarak tanımlayabiliriz.
- XML, HTML'ye bir alternatif olarak sunulmamıştır.

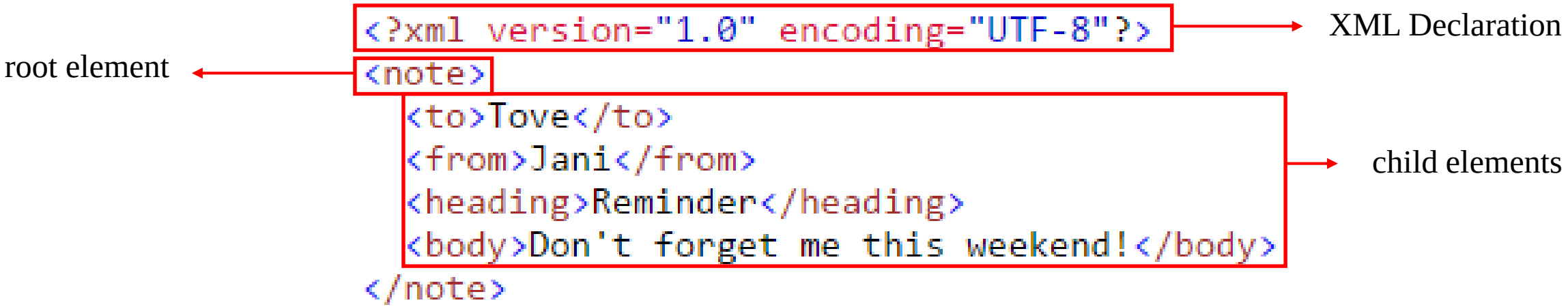
```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Kendi tanımladığımız bir xml

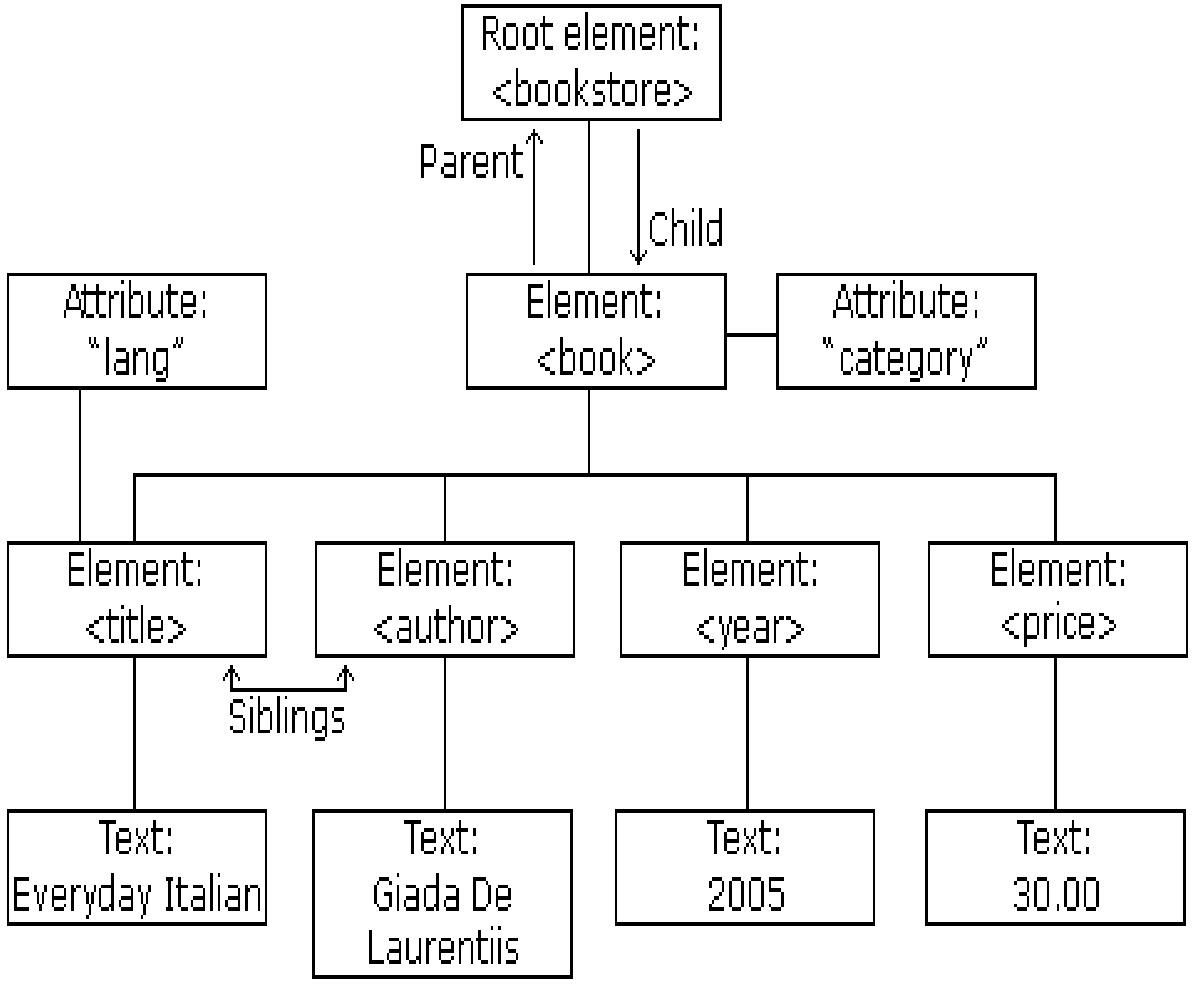
Kullanım

- Veriyi HTML'den ayırır.
- Veri paylaşımını basitleştirir.
- Veri transferini basitleştirir.
- Platform değişikliklerini basitleştirir.
- Verinin bulunabilirliğini arttırır.
- XHTML, XML Schema, WSDL gibi internet dilleri XML ile yazılmıştır.

XML Ağacı


<?xml version="1.0" encoding="UTF-8"?>
XML Declaration
root element
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
child elements
</note>

```xml
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</titl
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```



XML Ağacı

Kurallar

- Tüm xml elemanlarının bitiş taglarına sahip olması gerekir.
- Xml tagları karakter duyarlıdır. (CaseSensitive)
- Düzgün bir şekilde iç içe geçmelidirler.
- Bir tane root elemanı olmalıdır.
- Elemanların nitelikleri tırnak içerisine yerleştirilmelidir.

Xml Element & Attribute

```xml
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```xml
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

Xml Namespace

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

?

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Eğer, her iki kod parçası tek bir dosya içerisinde kullanılmaya çalışılırsa, <table> taglarında isim uyuşmazlığı oluşacaktır.

Önek (Prefix) Kullanarak Çözüm

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Yukarıdaki gibi bir kullanımda ise herhangi bir çakışma olmayacaktır.
<table> elemanları farklı isimlerde olacaktır.

- Xml dosyasında ön-ek (prefix) kullanıldığında bu ön-ek için sözde bir namespace tanımlanmalıdır.
- Namespace, başlangıç tagında xmlns attribute'ı olarak tanımlanır.

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

Namespace içerisinde yer alan URI bilgisi parser tarafından kullanılmaz.

URI'nin belirtilmesinin amacı namespace'e eşsiz (unique) bir isim vermektir.

- Bir eleman için varsayılan namespace tanımlandığı takdirde alt elemanlarında ayrıca tanımlamaya gerek yoktur.

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table xmlns="http://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

JSON
(JavaScript Object Notation)

- Veriyi saklamak ve transfer etmek üzere geliştirilmiş olan bir syntaxtır (http://www.w3schools.com/json/default.asp).
- XML'e bir alternatiftir. Kullanımı daha kolaydır.

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

```json
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

```xml
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JSON Object Creation in JavaScript</h2>

<p id="demo"></p>

<script>
var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}'

var obj = JSON.parse(text);

document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.street + "<br>" +
obj.phone;
</script>

</body>
</html>
```

XML ile Olan Benzerlikler

JSON ve XML'in ;

- Her ikisi de kendini tanımlayabilen bir yapıya sahiptir. Okunabilirdir.
- Her ikisi de hiyerarşik bir yapıya sahiptir.
- Her ikisi de bir çok programlama dili tarafından parse edilebilir.
- Her ikisi de XMLHttpRequest ile fetch edilebilir.

XML ile Olan Farklılıklar

- JSON bitiş tagı kullanmaz.
- JSON daha ile veri daha kısa bir şekilde ifade edilebilir.
- JSON okuma ve yazma işlemlerinde daha hızlıdır.
- JSON dizileri kullanabilir.

- En önemli farklılığı XML bir parser ile parse edilebilirken, JSON standart JavaScript fonksiyonları parse edilir.

JSON Syntax

- Javascript syntaxının bir alt kümesidir.
- Veri, name/value çiftleri halindedir.

```
"firstName":"John"
```

- Küme parantezleri "{ }" nesneleri belirtir.

```
{"firstName":"John", "lastName":"Doe"}
```

- Köşeli parantezler "[ ]" dizileri belirtir.

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

JSON - JavaScript

JSON objeleri oluşturulup JS objelerine atanabilirler.

```
var employees = [
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName": "Jones"}
];


// returns John Doe
employees[0].firstName + " " + employees[0].lastName;
```

- Dosya uzantısı ".json" 'dur.
- MIME tipi "application/json" dur.
- JSON nesneleri JS ile parse edilebileceği gibi jQuery ile de parse edilebilirler.

http://www.w3schools.com/js/default.asp

http://www.w3schools.com/dom/default.asp

http://www.w3schools.com/xml/default.asp

http://www.w3schools.com/json/default.asp