Introduction to Multimedia Technology

Dr. Mathias Lux
Alpen-Adria Universität Klagenfurt

First Of All: Today's Plan

- Whoami?
- Whoareyou?
- Todays schedule:
  - 9:00-10:15 part I
  - 10:30-11:30 part II
  - 13:00-14:00 DIY exercise
- Thursday
  - start at 9:00

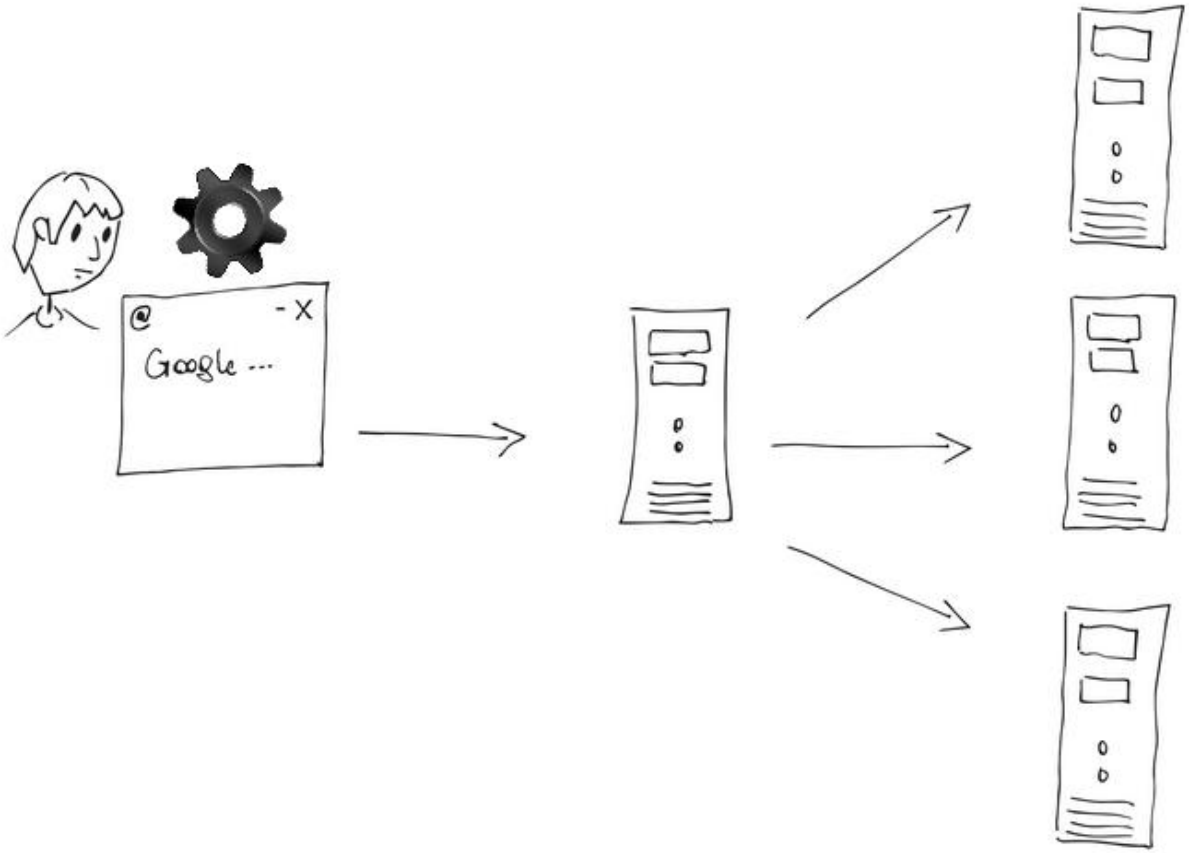
Master Plan - Topics

- Programming Primer
- JavaScript
- PHP
- SQL

Motivation: A Web Page

- Show an awesome web page and discuss how this is possible.

Motivation: How does it work?



Motivation: How does it work?

- Each component processes information
- User -> brain power
- Browser -> JavaScript
- Web server -> PHP
- Database -> SQL Query
- Web Services -> Requests
- Login Server -> Requests
- ...

Programming Languages: Statements

- Programming languages are based on statements (sentences)

```
var myNumber = 42;
```

- For each sentence we consider
  - Syntax: lexeme and tokens, structure
  - Semantics: static (at compile time) and dynamic (at run time)

Programming Languages: Symbols

- A symbol is a sequence of characters
- Symbols are defined by regular expressions


```
identifier = { letter | $ | _ }{ letter | $ | _ | digit }+
```

Programming Languages: Symbols

- Typical classes of symbols
- Identifier: myNumber, tmpFloat, …
- Literals: 42, “hello world”
- Keywords: while, if, else, for, …
- Special characters: <= > ; { }

Programming Languages

- Statements and symbols make up a program

- These programs are either
  - Compiled, or
  - Interpreted

Compiler

- A compiler transforms source code (text) to machine executable code.
- Basically that's zeros and ones.
- It takes care of "higher language constructs"
  - Machine languages are typically very simple
  - "Higher languages" offer convenient options for programmers
- Machine code is executed without knowledge on the source code.

Interpreter

- Source code is read by interpreter and executed.
- Interpreter manages storage, data, runtime environment.
- There is no intermediate step between writing source code and execution
  - ie. the code is not compiled.

Programming Languages: Constructs

- Variables
  - Assignment, blocks, scope …
- Conditions
- Control Structures
- Functions
- Data structures
- Objects

Variables

- Declaration

```
var myNumber;
```

- Assignment

```
myNumber = 42;
```

- Both at once

```
var myNumber = 42;
var myString = "Hello World!"
```

Variables: Dynamic Typing

```
var myNumber = 42;
var myString = "Hello World!"
myString = myString + 10
```

- Type of a variable is determined at runtime.

Assignments and Operators

- Using numbers

```
VarName = Expression;
```

- Adding, multiplying numbers

```
var myNumber = 42 * 3;
```

- Operating on Boolean variables

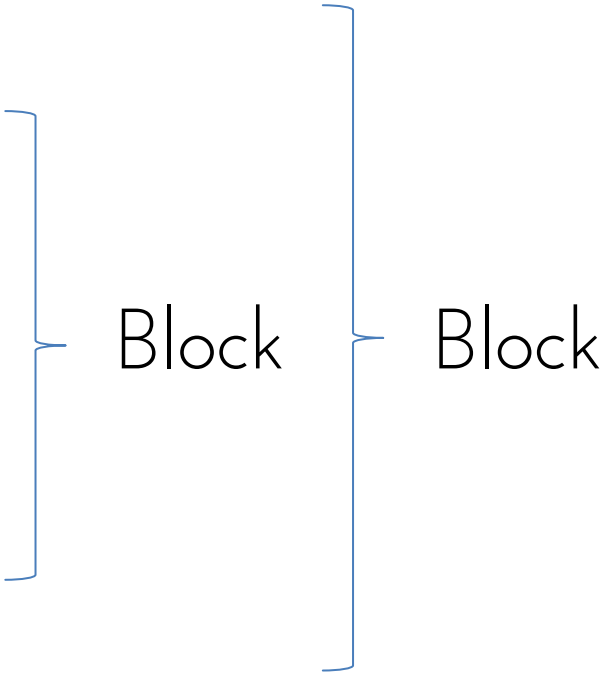
```
myBool = yes | no
```

- Using strings

```
myString = "Hello " + "World!"
```

Blocks

```
var myNumber = 42;
function squareIt() {
  var num = 8
  num = num*num;
  return num;
}
myNumber = squareIt();
```

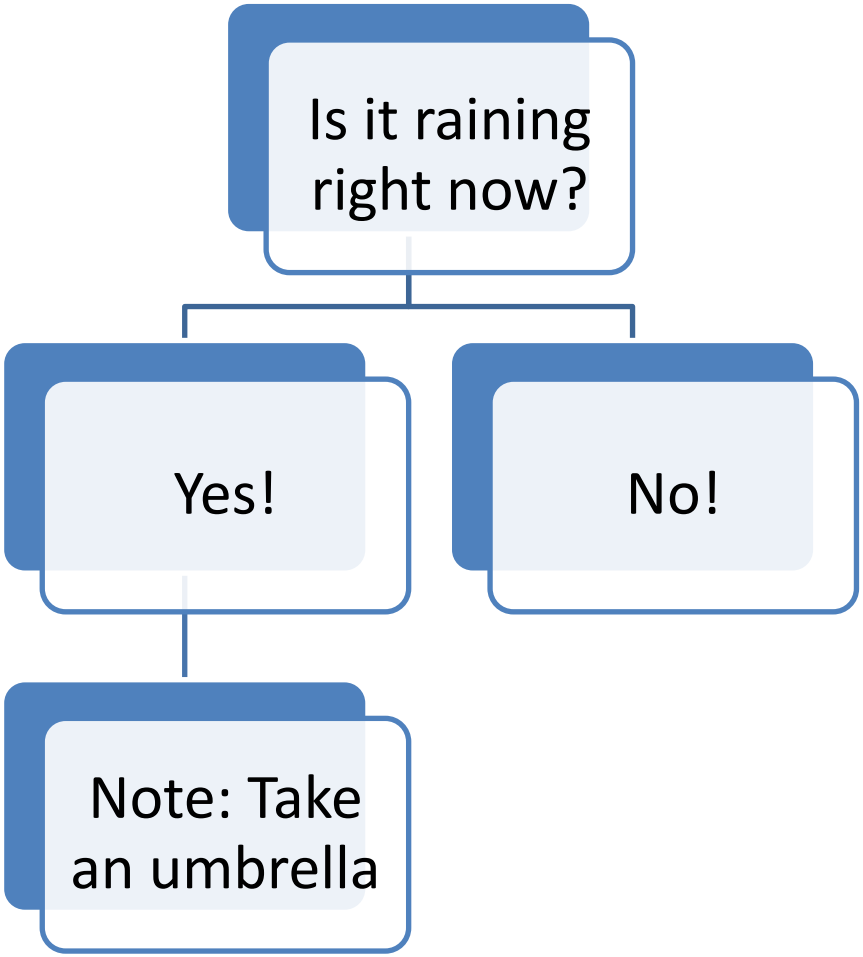


Scope

```
var myNumber = 42;
function squareIt() {
  var num = 8
  num = num*num;
  return num;
}
myNumber = num;
```

num is not defined in this scope!

Conditions



Conditions

```
var myNumber = 12;
if (myNumber < 10) {
    console.log("It's small!");
} else {
    console.log("It's big!");
}
```

Conditions: Switch ...

```
var myNumber = 2
switch(myNumber)
{
case 1:
  console.log("It's case 1!");
  break;
case 2:
  console.log("It's case 1!");
  break;
default:
  console.log("It's neither case 1 nor case 2!");
  break;
}
```

Conditions: "Short If"

```
var myNumber = 1;
var myString =
   myNumber>0?"positive":"negative";
console.log(myString);
```

Control Structures: Do ... While

```
var myNumber = 10;
do {
    myNumber -= 1;
} while (myNumber > 0)
console.log(myNumber);
// output is "0"
```

Control Structures: While ...

```
var myNumber = 10;
while (myNumber > 0) {
    myNumber -= 1;
}
console.log(myNumber);
// output is "0"
```

Control Structures: for ...

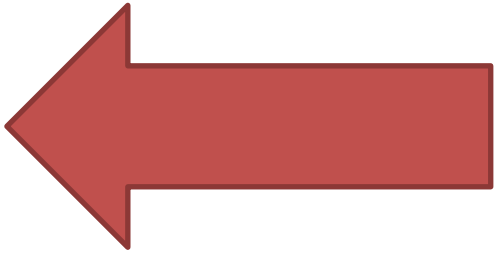
```
var myNumber = 10;
for (var i =0; i<10; i++) {
    myNumber -= 1;
}
console.log(myNumber);
// output is "0"
```

Functions

```
function doubleIt(snippet) {
    return snippet + " " + snippet;
}
var myText = "Ding!";
myText = doubleIt(myText)
console.log(myText);
```

Functions

```
function doubleIt(snippet) {
    return snippet + " " + snippet;
}
var myText = "Ding!";
console.log(doubleIt(myText));
```



Functions

```
function doubleIt(snippet) {
    return snippet + " " + snippet;
}
var myText = "Ding!";
console.log(doubleIt(doubleIt(myText)));
// What is the output?
```

Functions: Recursion

```
// Recursive function:
function half(param) {
    console.log(param);
    if (param > 1)
        return half(param/2);
    else
        return param;
}
var myNum = 20;
console.log("Result = " + half(myNum));
```

Data Types & Structures

- JavaScript hast the following data types
  - Number
  - String
  - Boolean
  - Null
  - Undefined
  - Object
- Data structure organize data in memory.

Data Types & Structures: Arrays

- Arrays are
  - sequences of instances
  - of (typically) a single data type
- Example in Java:

```
int[] arr = new int[]{1, 2, 3};
arr[0] = 3; //
arr[1] = arr[0] - arr[2];
```

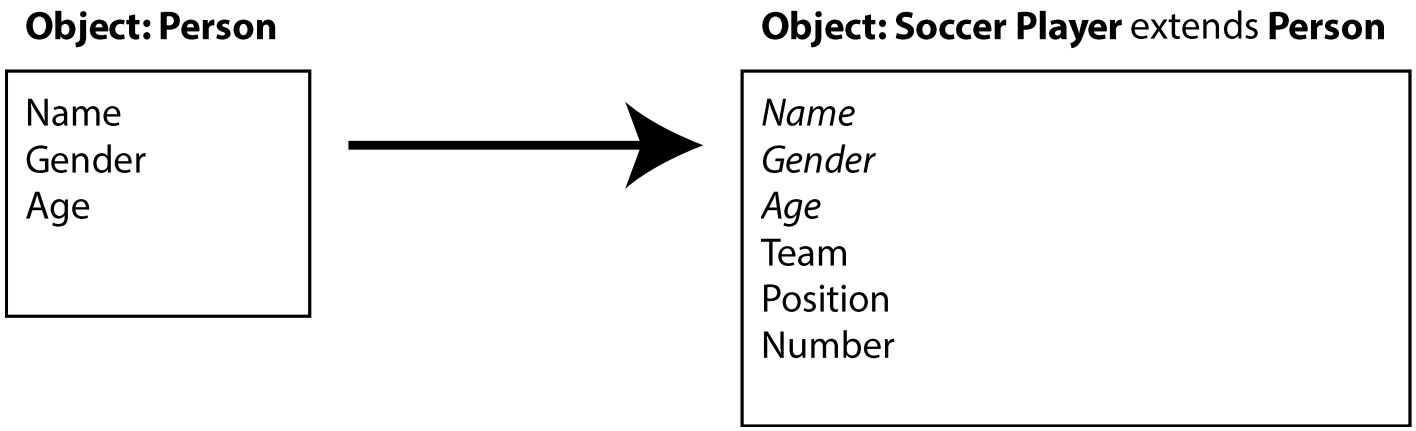
- Other data structures are …
  - Trees, lists, enumerations, …

Objects

- Objects are
  - Variables and data structure with
  - Manipulation methods (functions)
- Objects have members
  - Basic types
  - Other objects
  - Methods

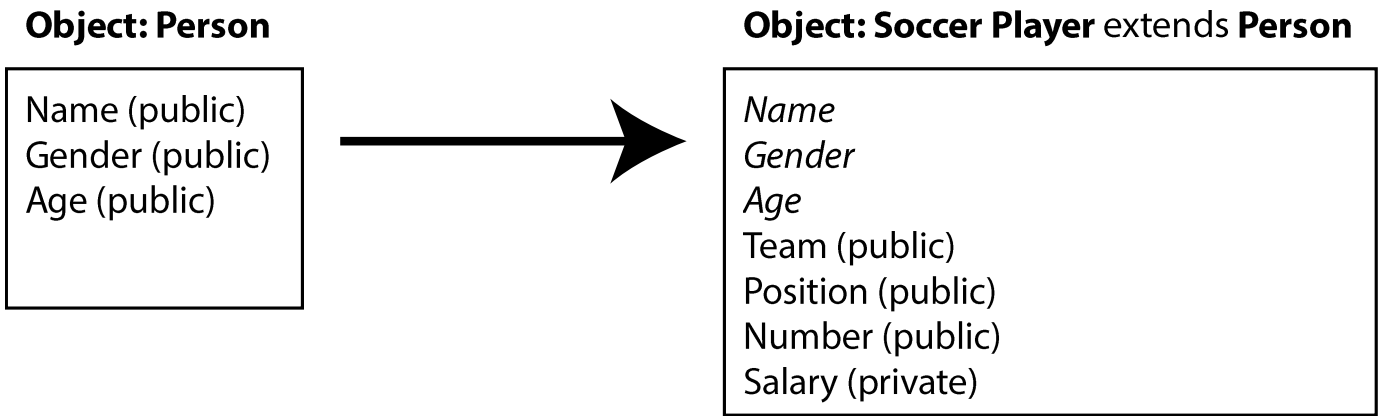
Objects: Main Ideas

- Inheritance
  - Objects can inherit from others



Objects: Main Ideas

- Restricted access
  - Access to members is controlled



JavaScript

- With JavaScript, you can write programs that are executed within the client's browser
  - Supported by all major browsers
  - It was designed to add interactivity to HTML pages
- A scripting language is a "light-weight" programming language
  - Interpreted (no compilation required)
  - Limited functionality compared, e.g., with Java
  - In general, except for syntax aspects, no commonalities with Java

JavaScript

- Common usage examples
  - Add text to HTML documents dynamically, change contents
  - React on events
  - Validate input data
  - Detect browser version, create cookies
- Libraries
  - Various libraries available (supporting Drag&Drop, Windows-like menus, tree control etc.)

JavaScript: History

- 1995: developed for Netscape 2
- 1996: Microsoft dialect JScript for Internet Explorer 3
- 1997: ECMAScript (specification handed in for standardization)
  - Ecma International: industry association for standardization of ICT
- 1999: ECMA-262 fourth edition

JavaScript: A First Example

```
<html>
<body>
   <script type="text/javascript">
        document.write("Hello World!");
        alert('Hello World');
   </script>
</body>
</html>
```

Placing JavaScript Code

- In the HTML head and body
  - code will be executed when sufficient content has been loaded
  - may be before whole page is loaded
- Functions
  - executed when they are called
- Attached to events of elements (executed when events are triggeres)
- Into an external file

JavaScript: Second Example

```
<h1 style="color:red;">I'm a header</h1>
<p style="color: green;" id="jsenabled">
   JS is disabled!
</p>
<script type="text/javascript">
   var node =
   document.getElementById("jsenabled");
   node.innerHTML = "JS is enabled";
</script>
```

JavaScript: Arrays

```
var myFruit = new Array();
myFruit[0] = "Pear";
myFruit[1] = "Cherry";
myFruit[2] = "Plum";
for (x in myFruit) {
    console.log(myFruit[x]);
}
```

JavaScript: Arrays

- Take a look at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

In-Course Exercise: Sort Array Descending

- Build a group
- Go to http://repl.it/
- Create an array of numbers or Strings
- Sort it in a descending manner

JavaScript: Strings

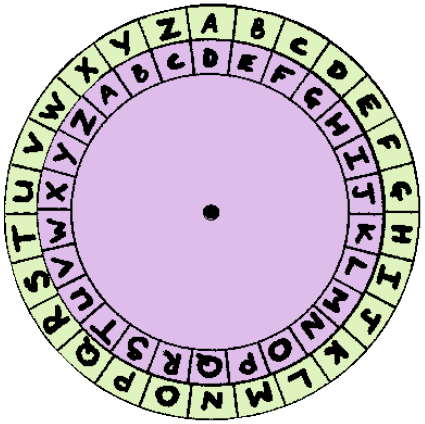
```
var myString = "media technology";
console.log(myString);
// prints media technology
console.log(myString.toUpperCase());
// prints MEDIA TECHNOLOGY
var myArr = myString.split(" ");
for (x in myArr) {
    console.log(myArr[x].slice(0,1).toUpperCase()
        + myArr[x].slice(1));
}
// prints Media Technology
```

JavaScript: Some Built-In Objects

- Math – cosine, random, ...
- Number – integer, float, NaN, infinity
- Date – access, compare, compute dates
- String – text processing
- RegExp – text processing
- Array – see above.

In-Course Exercise: Password Generator

- Create a JavaScript password generator
- 10 digits from 0-9

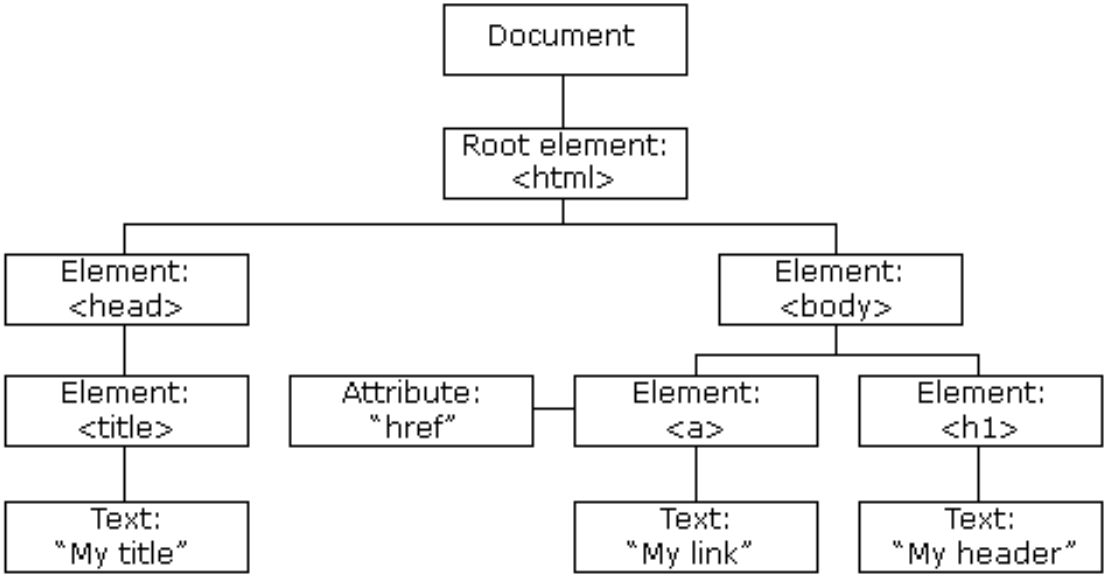


Possible Solution ..

```
function roll() {
    var dice = Math.random() * 6;
    dice = Math.floor(dice);
    // number to letter ?!?
    var letterArray = ["a", "b", "c", "d", "e", "f"];
    return (letterArray[dice])
}
var password ="";for (i=0;i<10;i++) {
    password = password + roll();
}
```

JavaScript: HTML DOM

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
  - see also http://www.w3schools.com/js/js_htmldom.asp



Using DOM JavaScript can ...

- change all
  - HTML elements in the page
  - HTML attributes in the page
  - CSS styles in the page
- remove existing HTML elements and attributes
- add new HTML elements and attributes
- react to all existing HTML events in the page
- create new HTML events in the page

DOM Example

```
<html>
<body>
<p id="intro">Hello World!</p>
<script>
txt=document.getElementById("intro").innerHTML;
document.write(txt);
</script>
</body>
</html>
```

JSON: JavaScript Object Notation

- Simple alternative to XML
- Creates a "tree of data"

- Investigate Example using
  - Chrome, F12, and console.write(…)

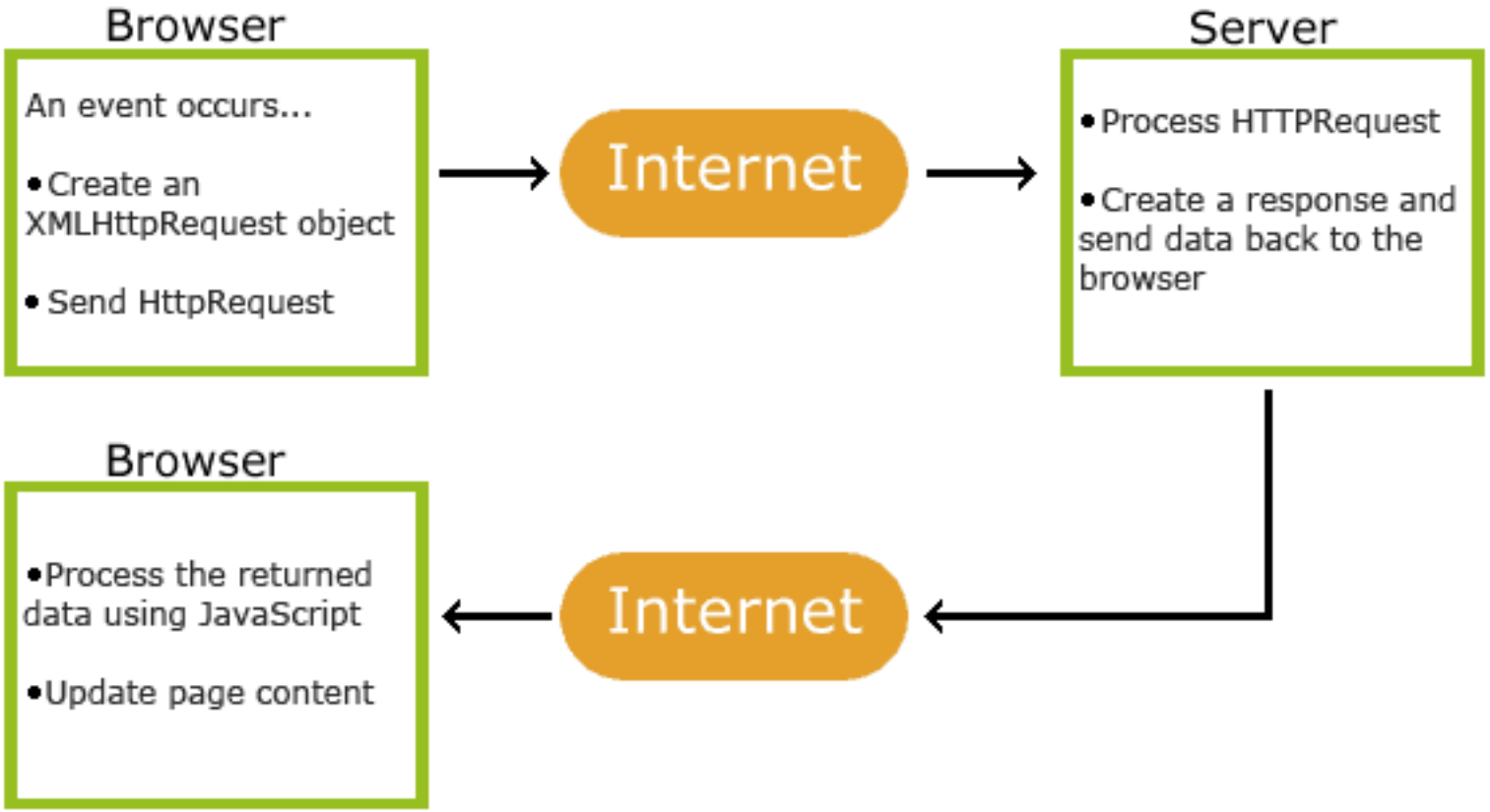
JavaScript: JSON

```
<script>
var myObj = {
    students:[
        {"name":"John",  "age":"32"},
        {"name":"Jane",  "age":"33"},
        {"name":"Jim",   "age":"34"},
        {"name":"Joana", "age":"35"}
    ]
}
console.log(myObj);
for (x in myObj.students) {
    document.write(myObj.students[x].name + "<br/>");
}
</script>
```

JavaScript: JSON

- For more information visit:
  - http://www.w3schools.com/json/

Asynchronous JavaScript and XML (AJAX)



src. http://www.w3schools.com/ajax/ajax_intro.asp

```
<script> function loadXMLDoc() {
        var xmlhttp;
        if (window.XMLHttpRequest) {
          xmlhttp=new XMLHttpRequest();
        } else {// code for IE6, IE5
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange=function() {
          if (xmlhttp.readyState==4 && xmlhttp.status==200){

        document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
          }
        }
        xmlhttp.open("GET","ajax_info.txt",true);
        xmlhttp.send();
    } </script>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
```

Asynchronous JavaScript and XML (AJAX)

- Draw sequence diagram ;)

AJAX: Examples

- Google Suggest
- YouTube
- Google Docs
- etc.

Example Library: JQuery

- A JavaScript library
- Tries to simplify JavaScript
- Provides a lot of enhanced methods
  - UI widgets, calls, common tasks, ...

Example Library: JQuery

- Import via CDN:

```
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js">
</script>
</head>
<body>
…
</body>
</html>
```

jQuery Syntax

Selecting elements -> performing action on the element(s).

- Basic syntax is: $(*selector*).*action*()
  - A $ sign to define/access jQuery
  - A (*selector*) to "query (or find)" HTML elements
  - A jQuery *action*() to be performed on the element(s)
- Examples:
  - $(this).hide() - hides the current element.
  - $("p").hide() - hides all <p> elements.
  - $(".test").hide() - hides all elements with class="test".
  - $("#test").hide() - hides the element with id="test".

jQuery Example

```
<html>
<head>
   <script src="jquery.min.js"></script>
</head>
<body>
<p>This is some text</p>
<a href='javascript:$("p").hide()'>click here to hide
   the paragraph</a>
</body>
</html>
```

jQuery

- For more information see
  - http://jquery.com/
  - http://www.w3schools.com/jquery/
- jQuery is currently de-facto standard.

jQuery UI

- See http://jqueryui.com/demos/

JavaScript Compression

- JavaScript is basically a long text file
- Compression makes this file smaller
  - to enhance download times
- JavaScript compression uses language semantics like
  - variable names, short ifs, consecutive declarations, remove blank spaces, ...

JavaScript Compression

- Try http://jscompress.com/

```
var myObj = {
  students:[
    {"name":"John",  "age":"32"},
    {"name":"Jane",  "age":"33"},
    {"name":"Jim",   "age":"34"},
    {"name":"Joana", "age":"35"}
  ]
}
console.log(myObj);
for (x in myObj.students) {
  document.write(myObj.students[x].name + "<br/>");
}
```

Compression Result ...

```
var myObj={students:[{name:"John",age:"32"},{name:"Jane",
age:"33"},{name:"Jim",age:"34"},{name:"Joana",age:"35"}]};
console.log(myObj);for(x in myObj.students){document.write(
myObj.students[x].name+"<br/>")}
```

Try it yourself …

- http://repl.it/
- http://www.w3schools.com/

What's Next?

- Server Side Scripting -> PHP
- Data Bases -> SQL
- And: how it all plays together ...

Exercise: Home Exercise

- Take the JavaScript Quiz:
  - http://www.w3schools.com/js/js_quiz.asp