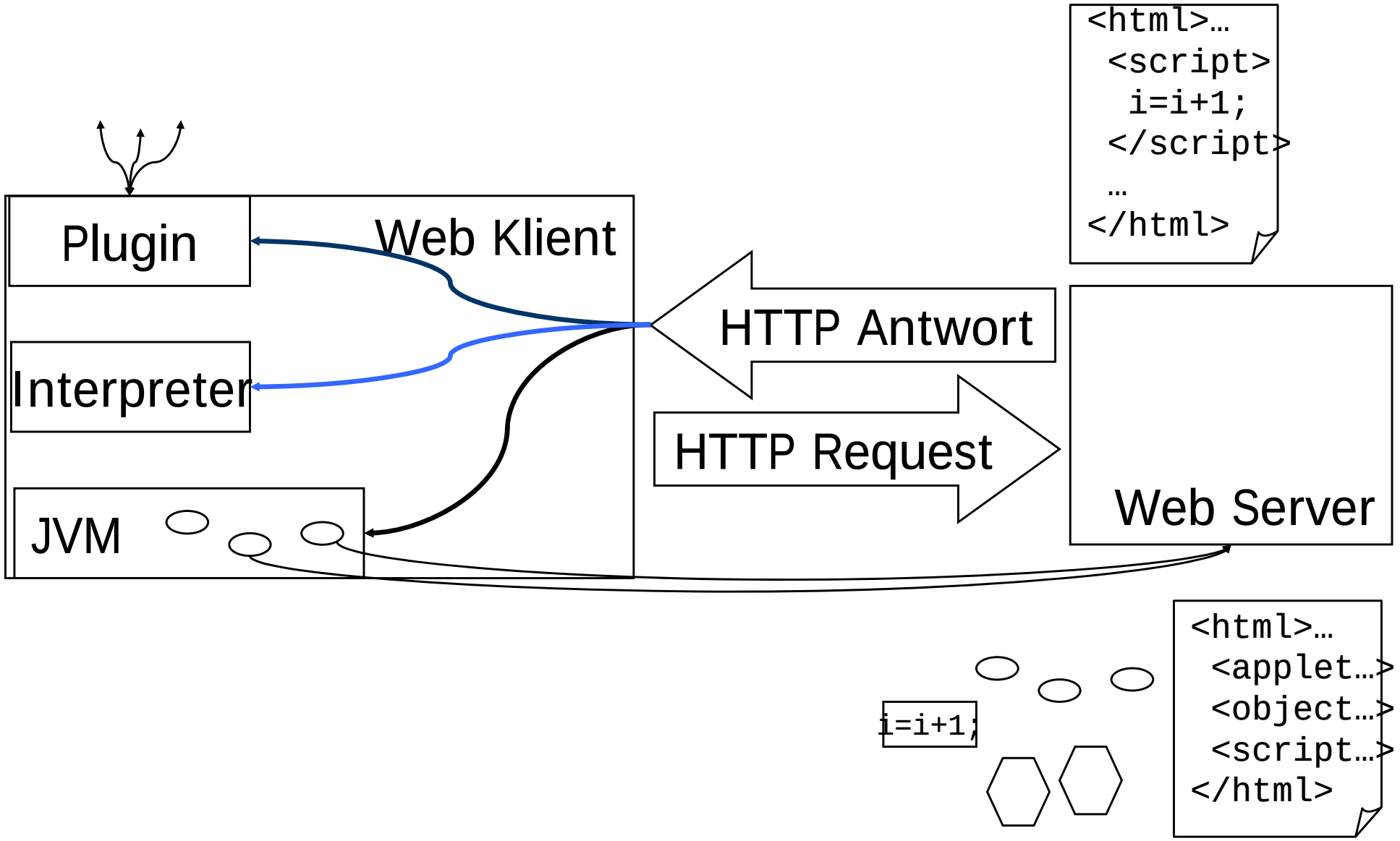
Netzprogrammierung
Klientenseitige Verarbeitung im Web

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

Überblick

1. Javascript
2. AJAX
3. Applets
4. Applets und RMI

Klientenseitige Verarbeitung



Javascript

JavaScript Entwicklung

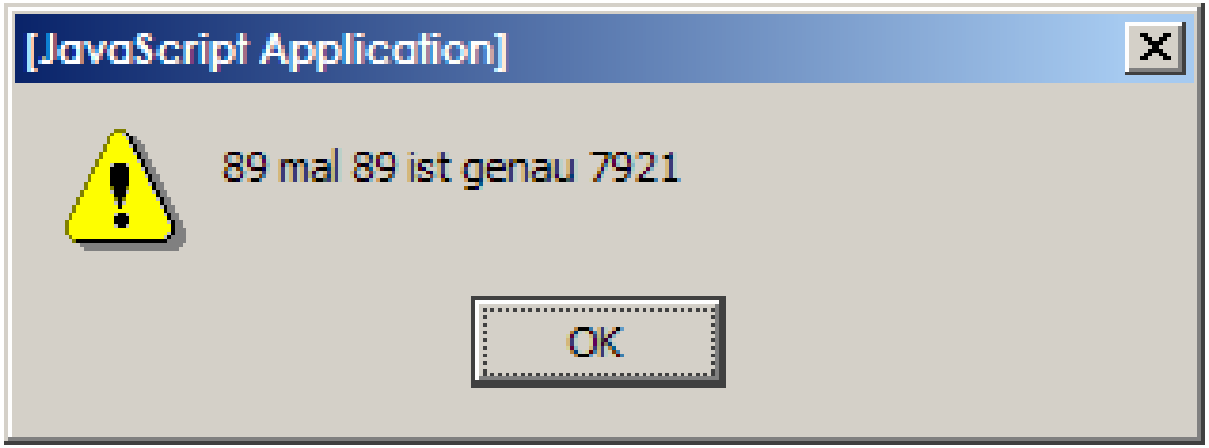
Freie Universität Berlin

- JavaScript: Einfache imperative Programmiersprache
  - Programmcode als Quelle
  - Eingebettet in HTML-Seite
  - Ausgeführt durch Interpreter im Browser
  - Zugriff auf Ereignisse und Dokumentenstruktur

- Meilensteine und Implementierungen:
  - JavaScript (1995): Netscape/Sun
  - ECMAScript: ECMA
  - JScript: Microsoft (JavaScript+Windows)
  - Seit Version 1.2 nicht mehr kompatibel in Browsern implementiert.
  - Seit 1.5: DOM Beachtung, uniforme Repräsentation des Dokumenteninhalts

Ausdrücke

- Vergleiche
  == != < <= => >
- Arithmetische Operatoren
  + - * / % ++ -- (+ auch Zeichenkettenkonkatenierung)
- Logische Operatoren
  && ||
- Bit-Operatoren
  >> << & | ^
- Beispiel:

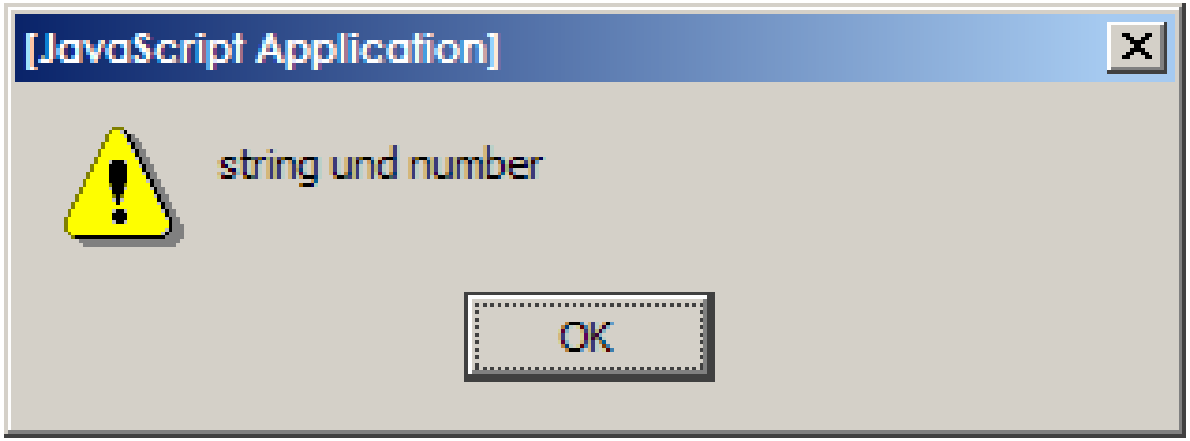



```
var i;
i=89;
var sqri;
sqri=i*i;
var message="89 mal 89 ist genau ";
alert(message + sqri);
```

Variablen

- Variablen sind schwach getypt
  - Müssen nicht als von einem Typen deklariert werden

    var i;
  - Haben je nach Inhalt einen Typen
  - Abfragbar mit typeof(*Ausdruck*):
    - boolean
    - string
    - number
    - function
    - object
    - undefined




```
alert(typeof(message) + " und " + typeof(sqri));
```

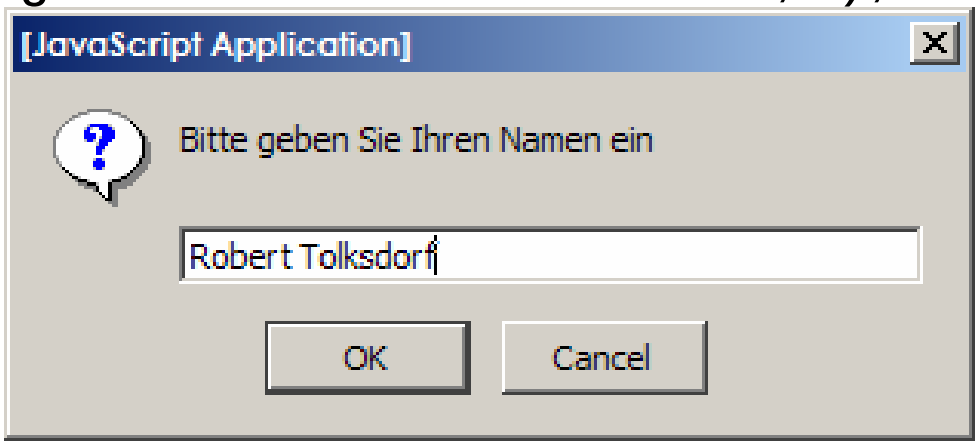
Felder

- Sind Objekte (siehe später) von Array
- Herkömmliche Felder
  - var Feld = new Array("rot", "grün", "blau");
  - var ersteFarbe = Feld[0];
- Assoziative Felder
  - var Feld = new Array();
  - Feld["PLZ"]="14195";
  - Feld["Ort"]="Berlin";
  - var stadt = Feld["Ort"];

Kontrollflusssteuerung

- Ein-/Zweifache Verzweigung

```
var name= window.prompt("Bitte geben Sie Ihren Namen ein","");
if (name=="") {
  alert("Sie Geheimniskrämer");
} else {
  alert("Hallo, "+name);
}
```



- Mehrfachverzweigung

```
var tag = window.prompt("Nummer des Wochentags?","");
switch(tag) {
 case "1": alert("Montag");
 break;
 case "2":  alert("Dienstag");
 break;
 ...
 default:  alert("Kann nicht sein");
 break;
}
```

Schleifen

- Abweisend

```
while (i<=n) {
  fac=fac*i;
  i++;
}
```

- Nichtabweisend

```
do {
  fac=fac*i;
  i++;
} while (i<=n);
```

- Zählschleife

```
for (var i=1; i<=n; i++)
  fac=fac*i;
```

- Verlassen des Schleifenkörpers und Neuiteration:

```
continue;
```

- Verlassen der aktuellen Schleife:

```
break;
```

Funktionen

- Deklaration mit Parametern, lokalen Variablen und Rückgabewert
- Beispiel

```
function ffac(n) {
  var fac=1;
  for (var i=1; i<=n; i++)
    fac=fac*i;
  return fac;
}
```

- Aufruf als Ausdruck

```
var fac5=ffac(5);
```

Skripte einbinden

Freie Universität Berlin

- Direkt in HTML

```
<script type="text/javascript">
<!--
function ffac(n) {
  var fac=1;
  for (var i=1; i<=n; i++)
    fac=fac*i;
  return fac;
}
alert(ffac(5));
// -->
</script>
```

- Extern in eigener Datei

```
<html>
<head>...
<script src="fac.js" type="text/javascript">
</script>
</head>...
```

Objekte

- „Konstruktor“ als Funktion definieren

```
function Ort(PLZ, Name) {
  this.PLZ=PLZ;
  this.Name=Name;
  this.Anschrift=PLZ+" "+Name;
}
```

- Exemplare erzeugen

```
var FUOrt=new Ort("14195","Berlin");
var TUOrt=new Ort("10578","Berlin");
```

- Felder selektieren

```
alert(FUOrt.Anschrift);
```

Vordefinierte Objekte

- In JavaScript sind verschiedene Objekte eingebaut, insbesondere die Repräsentation dss Dokuments im Browser

```
<html>
<body>
<h1>Startseite</h1>
<script type="text/javascript">
<!--
var name= window.prompt("Bitte geben Sie Ihren Namen ein","");
if (name=="") {
  document.write("<b>Sie Geheimniskrämer</b>");
} else {
  document.write("Hallo, "+name);
}
// -->
</script>
<p>Jetzt aber los...</p>
```

document Objekt

- Eigenschaften:
  - alinkColor (Farbe für Verweise beim Anklicken)
  - bgColor (Hintergrundfarbe)
  - charset (verwendeter Zeichensatz)
  - cookie (beim Anwender speicherbare Zeichenkette)
  - defaultCharset (normaler Zeichensatz)
  - fgColor (Farbe für Text)
  - lastModified (letzte Änderung am Dokument)
  - linkColor (Farbe für Verweise)
  - referrer (zuvor besuchte Seite)
  - title (Titel der Datei)
  - URL (URL-Adresse der Datei)
  - vlinkColor (Farbe für Verweise zu besuchten Zielen)
- Normal benutzen:

```
<p>Gehen Sie dahin wo Sie herkommen:
<script type="text/javascript">
<!--
document.write(document.referrer);
//-->
</script>
</p>
```

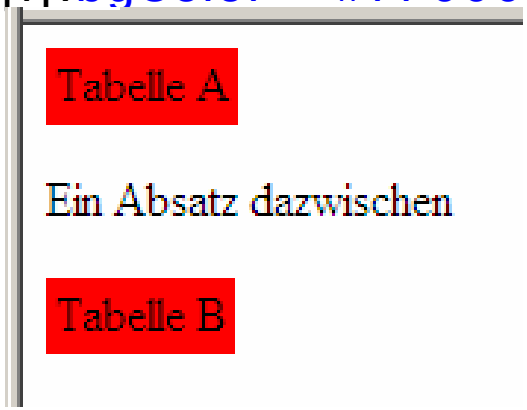
Objektmodell des Browsers

- Unter document sind alle Elemente einer HTML-Seite im Browser auffindbar:
  - document.getElementsByTagName("table")
    - Liefert ein Feld mit allen <table> Elementen
  - document.getElementsById("Adresse")
    - Liefert das Element mit dem Attribut id="Adresse"
  - document.getElementsByName("Adresse")
    - Liefert das Element mit dem Attribut name="Adresse„
- Von document aus Zugang zu Fenstern, History, Browsereigenschaften, Plugins, Bildschirmeigenschaften etc.

Eigenschaften

- Eigenschaften sind teilweise änderbar (abhängig von JavaScript Version und Implementierung...)

```
<body>
<table><tr><td>Tabelle A</td></tr></table>
<p>Ein Absatz dazwischen</p>
<table><tr><td>Tabelle B</td></tr></table>
<script type="text/javascript">
<!--
for (var i=0; i<document.getElementsByTagName("table").length; i++)
   {
 document.getElementsByTagName("table")[i].bgColor="#FF0000";
}
// -->
</script>
```



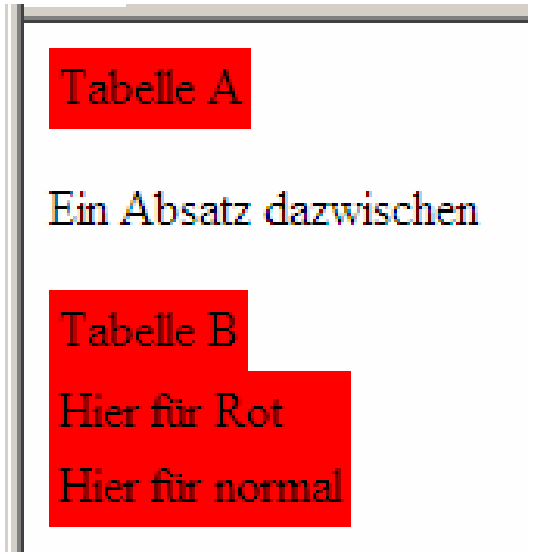
Ereignisse

- JavaScript-Funktionen können als Ereignisbehandlung in HTML notiert werden
- Mitteilung beim Laden der Seite ausgeben:
  - <body onLoad="alert('Guten Tag')">
- Definierte HTML-Attribute für Elemente:
  - onAbort (Abbruch)
  - onBlur (Verlassen)
  - onChange (bei Änderung), onError (im Fehlerfall), onFocus (Aktivieren)
  - onClick (bei Klicken), onDblClick (bei Doppelklick)
  - onKeydown (gedrückte Taste), onKeypress (gedrückt gehaltene Taste), onKeyup (losgelassene Taste)
  - onLoad (Laden der Seite), onUnload (Verlassen der Seite)
  - onMousedown (gedrückte Maustaste), onMousemove (weiterbewegte Maus), onMouseout(Verlassen des Elements mit der Maus), onMouseover (Überfahren des Elements mit der Maus), onMouseUp (bei losgelassener Maustaste)
  - onSelect (Selektieren von Text), onReset (Zurücksetzen des Formulars), onSubmit (Absenden des Formulars)

Tabellenzellen ändern Farbe aller Tabellen

Freie Universität Berlin

```
<script type="text/javascript">
<!--
function colorTables(color) {
  for (var i=0; i<document.getElementsByTagName("table").length; i++) {
   document.getElementsByTagName("table")[i].bgColor=color;
  }
}
// -->
</script>
<table><tr><td>Tabelle A</td></tr></table>
<p>Ein Absatz dazwischen</p>
<table><tr><td>Tabelle B</td></tr></table>
<table>
 <tr><td onMouseover='colorTables("#FF0000")'>Hier für Rot</td></tr>
 <tr><td onMouseover='colorTables("")'>Hier für normal</td></tr>
</table>
```



Überprüfen aller Eingabefelder auf Eingaben [selfhtml]

Freie Universität Berlin

```
[...]<script type="text/javascript">
<!--
function CheckInput() {
 for(i=0; i<document.forms[0].elements.length; ++i)
  if(document.forms[0].elements[i].value == "")  {
    alert("Es wurden nicht alle Felder ausgefuellt!");
        document.forms[0].elements[i].focus();
        return false;
   }
  return true;
}
//-->
</script></head><body>
<form action="onsubmit.htm" onSubmit="return CheckInput();">
Feld 1: <input size="30"><br>Feld 2: <input size="30"><br>
Feld 3: <input size="30"><br>
<input type="submit">
</form>[...]
```

JavaScript

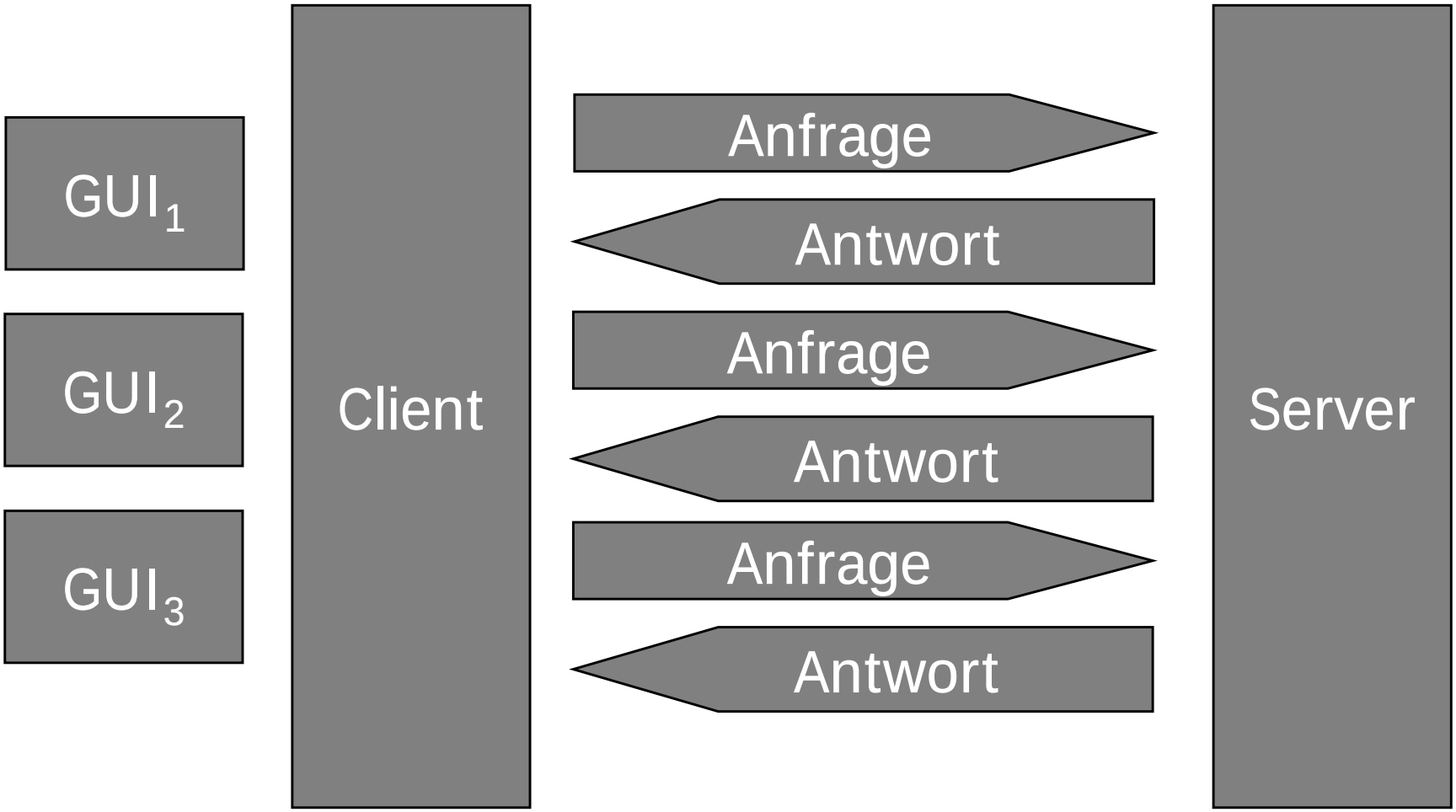
- Vollständige Sprachbeschreibung umfangreich
- Objektreferenz entscheidend
- Technisch nicht trivial wg. Inkompatibilitäten

- SelfHTML/JavaScript bei http://de.selfhtml.org/javascript/index.htm ist kostenlosen sehr gute Online-Referenz

Asynchronous JavaScript and XML (AJAX)

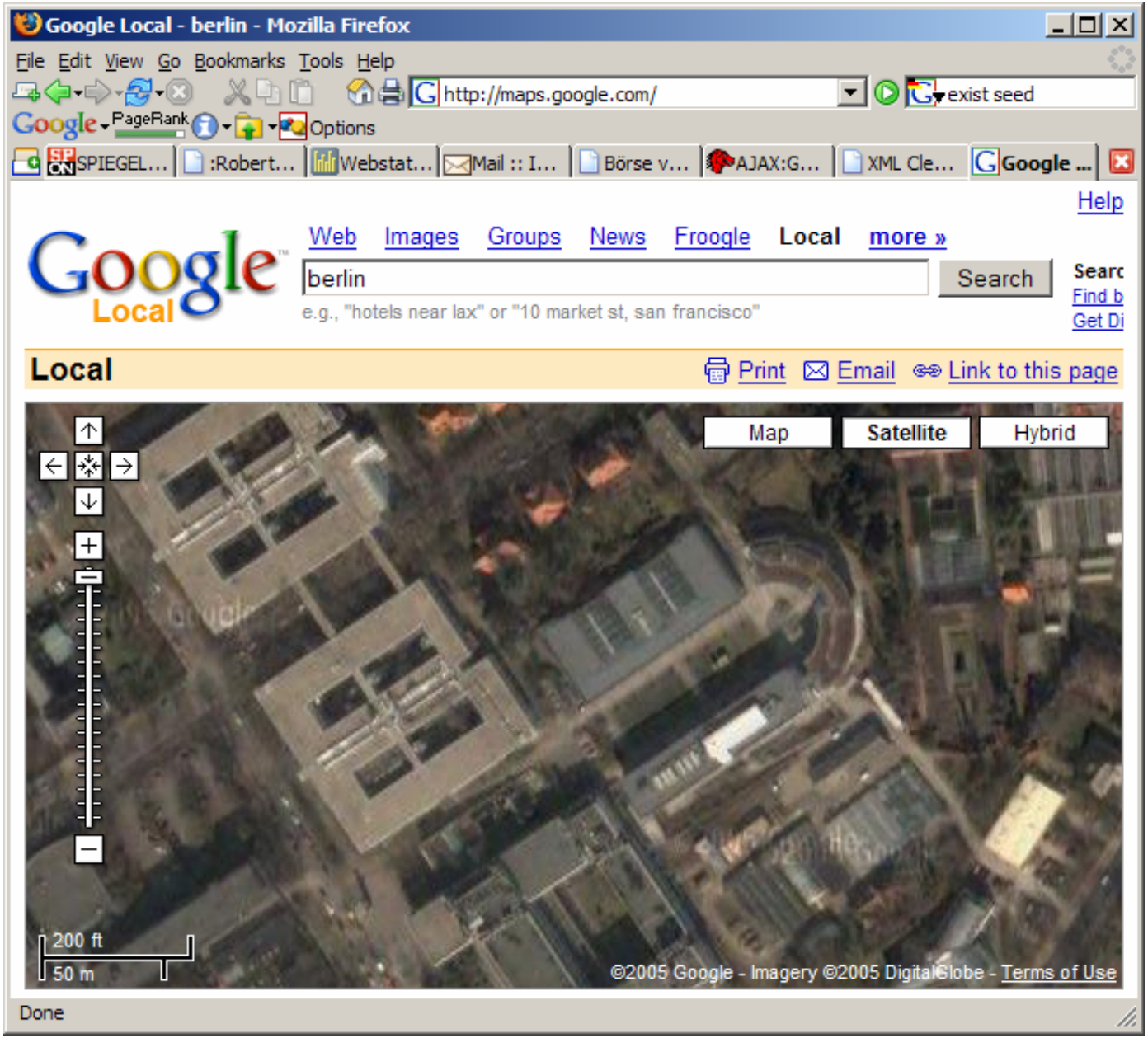
Freie Universität Berlin

- Problematik: Durch die Anfrage/Antwort Interaktion per HTTP wird bei jeder Nutzeraktion ein komplett neues GUI im Browser erzeugt



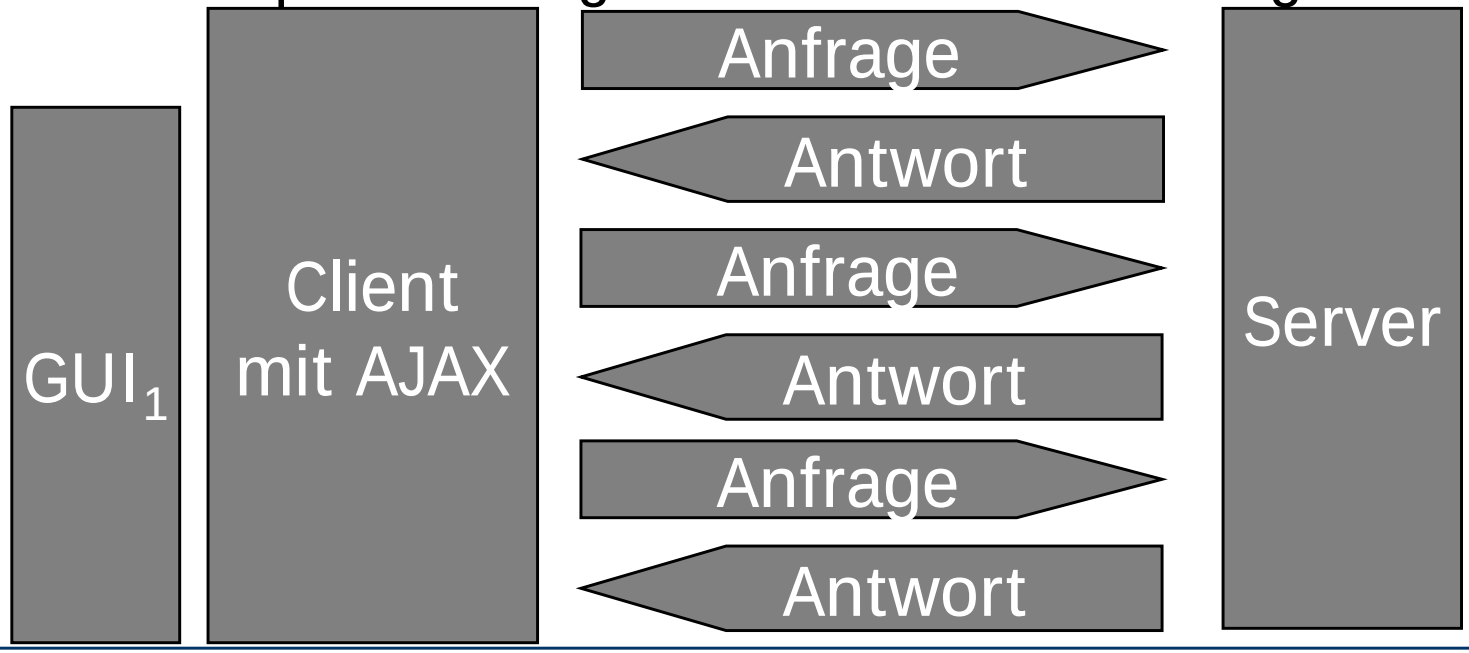
Änderbares GUI

- Ziel:
  Nur Teile des GUIs sollen nach Nutzer-interaktion mit nach-geladenen Daten geändert werden



AJAX

- Asynchronous JavaScript and XML (AJAX) realisiert dies durch Kombination von
  - Präsentationssprachen XHTML und CSS
  - Interaktion und Modifikation im Browser mit DOM
  - Datenaustausch mit XML
  - Datentransfer durch asynchrone HTTP-Anfragen
  - Javascript als Integration dieser Technologien



AJAX nutzen (nach Mozilla Developer Center. AJAX:Getting Started)

Freie Universität Berlin

- JavaScript-Aufruf an HTML-Element binden

```
<img src="arrow.gif" onclick="loadNewData();">
```

- Feststellen, in welcher Javascript-Umgebung wir sind und ein „Objekt“ für einen HTTP-Zugriff erstellen:

```
if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- Beim http_request-Objekt eine JavaScript Methode binden, die bei Zustandsänderung aufgerufen wird

```
http_request.onreadystatechange = function(){
    // do the thing
};
```

AJAX nutzen (nach Mozilla Developer Center. AJAX:Getting Started)

- Asynchrone HTTP-Anfrage starten

```
http_request.open('GET',
                  'http://www.example.org/some.file', true);
http_request.send(null);
```

  - Es kann auch ein Query-String bei send geschickt werden
  - Es kann auch POST verwendet werden
- Zustand der Anfrage
  - http_request.readyState zwischen 0 und 4
  - Jedesmal wird der registrierte Callback aufgerufen
  - Bei Zustand 4 ist der Zugriff beendet

```
if (http_request.readyState == 4) {
    // everything is good, the response is received
} else {
    // still not ready
}
```

- HTTP-Statuscode prüfen

```
if (http_request.status == 200) {
    // perfect!
} else {
    // there was a problem with the request,
    // for example the response may be a 404 (Not Found)
    // or 500 (Internal Server Error) response codes
}
```

AJAX nutzen (nach Mozilla Developer Center. AJAX:Getting Started)

Freie Universität Berlin

```
<script type="text/javascript" language="javascript">
   var http_request = false;
   function makeRequest(url) {
      http_request = false;
      if (window.XMLHttpRequest) { // Mozilla, Safari,...
         http_request = new XMLHttpRequest();
         if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
            // See note below about this line
         }
      } else if (window.ActiveXObject) { // IE
         try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
         } catch (e) {
            try {
               http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
         }
      }
      if (!http_request) {
         alert('Giving up :( Cannot create an XMLHTTP instance');
         return false;
      }
      http_request.onreadystatechange = alertContents;
      http_request.open('GET', url, true);
      http_request.send(null);
   }
```

```
    function alertContents() {
        if (http_request.readyState == 4) {
            if (http_request.status == 200) {
                alert(http_request.responseText);
            } else {
                alert('There was a problem with the request.');
            }
        }

    }
</script>
<span
    style="cursor: pointer; text-decoration: underline"
    onclick="makeRequest('test.html')">
        Make a request
</span>
```

- test.xml:

```
<?xml version="1.0" ?>
<root>
    I'm a test.
</root>
```

- Im HTML:

```
...
onclick="makeRequest('test.xml')">
...
```

- Verarbeitung:

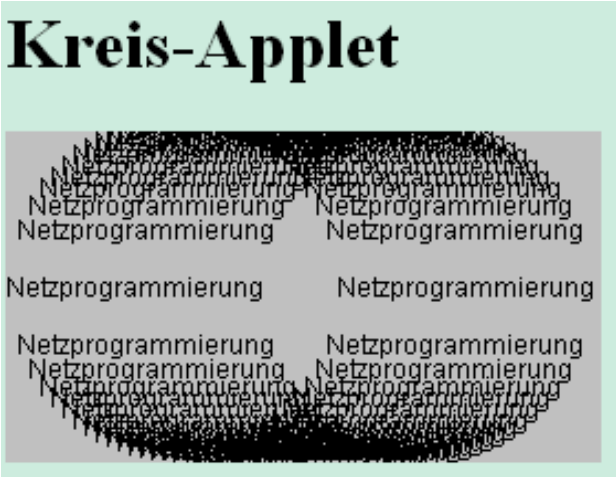
```
var xmldoc = http_request.responseXML;
var root_node =
    xmldoc.getElementsByTagName('root').item(0);
alert(root_node.firstChild.data);
```

Applets

Applets

- Applets sind (kleinere) Java-Programme, die in einem Java-fähigen Web-Browser gestartet werden können.
- Das Einbinden von Applets in eine HTML-Seite erfolgt mit dem <applet>-Tag.
- Alle Applets sind Unterklassen von java.applet.Applet
- Die Klasse Applet hat folgende Oberklassen
  - java.lang.Object
  - java.awt.Component
  - java.awt.Container
  - java.awt.Panel

Beispiel für ein <applet>-Tag




```
<applet
        codebase="."
        code="Kreis.class"
        width="270"
        height="150"
        alt="Das Kreis-Applet läuft leider nicht!">
<param name="string" value="Netzprogrammierung">
<param name="radius" value="75">
<param name="schritt" value="5">
<p>
  Hier sollte eigentlich ein <b>Kreis-Applet</b> dargestellt
   werden, aber ihr Browser versteht leider keine Applets.
</p>
</applet>
```

Einbinden von Applets in eine HTML-Seite (1)

Freie Universität Berlin

- mit dem <applet>-Tag

```
<applet
[ codebase="applet-url" ] [ archive="archiv-liste" ]
code="applet-dateiname" oder object="serialisiertes-applet"
width="breite" height="höhe"
[ alt="alternativer-text" ] [ name="applet-instanz-name" ]
[ align="ausrichtung" ]
[vspace="vertikaler-abstand"]
[hspace="horizontaler-abstand"]>
[ <param name="parameter0" value="wert0"> ]
[ < param name ="parameter1" value ="wert1"> ]
...
[ < param name ="parameterN" value ="wertN"> ]
[ html-text ]
</applet>
```

Parameter des <applet>-Tags (1)

Freie Universität Berlin

- codebase (optional)
  - Verzeichnis (URL), in dem die Appletdatei steht
  - falls nicht vorhanden: Verzeichnis der HTML-Seite (URL)
- archive (optional)
  - Liste mit einem oder mehr Archiven, die Klassen enthalten, welche initial geladen werden. Die Archive werden durch Kommata getrennt.
- code
  - class-Datei des Applets relativ zu codebase
- object
  - Name einer Datei, die eine serialisierte Repräsentation des Applets enthält. Das Applet wird deserialisiert. Die Methode init() wird nicht aufgerufen, sondern nur die Methode start().

Parameter des <applet>-Tags (2)

- width anfängliche Breite des Applets in Pixel
- height anfängliche Höhe des Applets in Pixel
- alt (optional)
  - Text, der angezeigt werden soll, wenn der Browser das <applet>-Tag kennt, aber das Applet nicht läuft, oder wenn der Mauszeiger über dem Applet steht
- name (optional)
  - Name der Instanz eines Applets für Kommunikation zwischen Applets
- align (optional)
  - Ausrichtung des Applets auf der WWW-Seite (vgl. <img>-Tag)
  - mögliche Werte: top, texttop, middle, absmiddle, bottom, absbottom

Parameter des <applet>-Tags (3)

Freie Universität Berlin

- vspace (optional)
  - Breite des Randes ober- und unterhalb eines Applets in Pixel
- vspace (optional)
  - Breite des Randes links und rechts eines Applets in Pixel
- <param>-Tag (optional)
  - Parameter, die an ein Applet übergeben werden
  - Name und Wert sind Strings
- HTML-Text (optional)
  - wird durch nicht Java-fähige WWW-Browser angezeigt

Methoden in Applets

- Zum Starten eines Applets werden von einem Browser oder Appletviewer bestimmte Methoden aufgerufen (ähnlich der Einstiegsmethode main() in Anwendungen).
- Diese Methoden sind in der Klasse Applet (vor)definiert und müssen in eigenen Applets überschrieben werden.

Methoden für alle Applets (1)

Freie Universität Berlin

- public void init()
  - wird vom WWW-Browser vor dem Start aufgerufen
  - zur Initialisierung (z.B. Zeichensatz, Vorder-/Hintergrundfarbe)
  - Einlesen von Parametern (mit Methode getParameter(), s.u.)
  - überschreibt Methode aus der Klasse java.applet.Applet
- public void paint(Graphics g)
  - wird vom WWW-Browser zum Start aufgerufen
  - das Graphics-Objekt stellt der WWW-Browser zur Verfügung
  - stellt den Inhalt des Applets dar
  - überschreibt Methode aus der Klasse java.awt.Component
- public String getParameter(String param)
  - gibt den String-Wert eines Applet-Parameters zurück (vgl. <param name="parameter" value="wert">-Tag in <applet>)
  - definiert in der Klasse java.applet.Applet

Methoden für alle Applets (2)

- public String[][] getParameterInfo()
  - stellt Informationen zu den Parametern eines Applets bereit. Der Rückgabewert ist ein Array mit Array mit drei Strings:
    - Parametername
    - Parametertyp
    - Parameterbeschreibung
  - überschreibt Methode aus der Klasse java.applet.Applet
- public String getAppletInfo()
  - stellt allgemeine Informationen über das Applet zur Verfügung (AutorInnen, Bedienung, Implementierung usw.)
  - überschreibt Methode aus der Klasse java.applet.Applet
- public void showStatus(String msg)
  - gibt eine Nachricht in der Statuszeile des WWW-Browsers aus
  - definiert in der Klasse java.applet.Applet

Die Datei Kreis.java (1)

Freie Universität Berlin

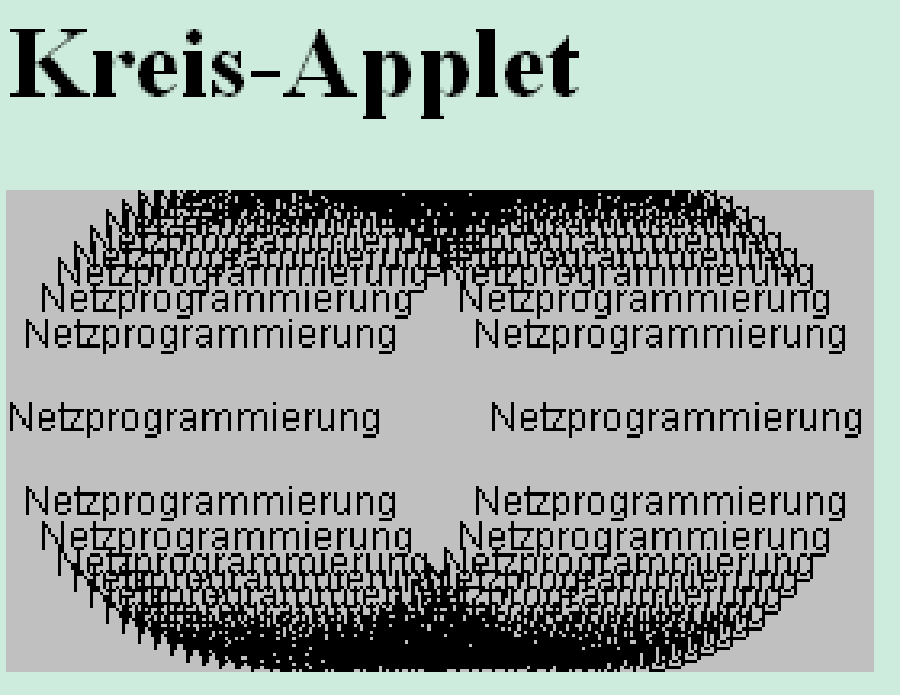
```
import java.applet.*; // für Applets
import java.awt.*;    // für Graphiken

public class Kreis extends Applet {

  private String str;
  private int r,schritt;

  public void init() {
    str = getParameter("string");
    r = getIntParameter("radius");
    schritt = getIntParameter("schritt");
  } // Ende Methode init()

  protected int getIntParameter(String name) {
    try {
      return Integer.parseInt(getParameter(name));
    }
    catch (NumberFormatException e) {
      return 1;
    }
  } // Ende Methode getIntParameter()
```

Die Datei Kreis.java (2)

```
  public void paint(Graphics g) {
    try { g.setColor(Color.black);
      for (int x=-r; x<r+1; x=x+schritt) {
        int y = (int)Math.sqrt(Math.pow(r,2)-Math.pow(x,2));
        g.drawString(str,x+r,y+r);
        g.drawString(str,x+r,-y+r);
      }
    }
    catch (NullPointerException e) { // "string"-Parameter fehlte
      str = "."; paint(g);
    }
  } // Ende Methode paint()
  public String[][] getParameterInfo() {
    String[][] info = {
      {"string",  "String Wert", "Ausgabestring"},
      {"radius",  "int Wert",    "Radius in Pixel"},
      {"schritt", "int Wert",    "Ausgabeschritte in Pixel"}
    };
    return info;
  } // Ende Methode getParameterInfo()
  public String getAppletInfo() {
    return "String-Kreis-Applet\n"+"von Wilhelm Weisweber(ww@cs.tu-berlin.de)";
  } // Ende Methode getAppletInfo()
} // Ende Klasse Kreis
```

Beispiel: Einfaches Applet

```
<h1>Kreis-Applet</h1>
<applet
   codebase="." code="Kreis.class"
   width="270" height="150"
   alt="Das Kreis-Applet läuft leider nicht!">
<param name="string"
   value="Netzprogrammierung">
<param name="radius" value="75">
<param name="schritt" value="5">
<p>Hier sollte eigentlich
   ein <b>Kreis-Applet</b>
   dargestellt werden, aber ihr Browser
   versteht leider keine Applets.</p>
</applet>
```



Applets mit Interaktion

Applets mit Interaktion

- Die Klasse Applet ist eine mittelbare Unterklasse der Klasse Component.
- Alle Methoden der Klasse Component zur Bearbeitung von Ereignissen stehen zur Verfügung:
  - zum Überschreiben (pro Ereignisklasse XYZEvent)
    - protected void processEvent(AWTEvent)
    - protected void processXYZEvent(XYZEvent)
  - zum Aufrufen (pro Ereignisklasse XYZEvent)
    - public void addXYZListener(XYZListener)
    - public void removeXYZListener(XYZListener)
    - public void enableEvents(long maske)
    - public void disableEvents(long maske)

Beispiel: Interaktives Applet

```
<h1>Interaktives Applet</h1>
<applet
   codebase="."
   code="Interaktion.class"
   width="200" height="100"
   alt="Das interaktive Applet
        läuft leider nicht!">
<p>Hier sollte eigentlich ein
   <b>interaktives Applet</b>
   dargestellt werden, aber ihr
   Browser versteht leider
   keine Applets.</p>
</applet>
```

Die Datei Interaktion.java (1)

Freie Universität Berlin

```
import java.applet.*;    // fuer Applets
import java.awt.*;       // fuer Graphiken
import java.awt.event.*; // fuer Ereignisse

public class Interaktion extends Applet {
  private TextField text = new TextField();
  private Button anzeigen = new Button("Anzeigen");
  public void init() {
    this.setLayout(new GridLayout(2,1));
    Rectangle r = this.getBounds();
    this.setFont(new Font("Helvetica", Font.PLAIN, r.height/4));
    ActionListener aktion = new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        ausgabe(e.getSource());
      }
    };
    anzeigen.addActionListener(aktion);
    this.add(text);
    this.add(anzeigen);
  } // Ende Methode init()
```

Die Datei Interaktion.java (2)

Freie Universität Berlin

```
  public void ausgabe(Object komponente) {
    if (komponente==anzeigen)
      showStatus(text.getText());
  }

  public String[][] getParameterInfo() {
    String[][] info = {{}}; // kein Parameter
    return info;
  }

  public String getAppletInfo() {
    return "Interaktives Applet\n"+
           "Von Wilhelm Weisweber (ww@cs.tu-berlin.de)";
  }

} // Ende der Klasse Interaktion
```

Applets und RMI

RMI Zugriff aus Applets heraus

- Applets können auch auf RMI Objekte zugreifen
- Interaktion wie beim normalen RMI
- Zugriff in der Regel aber auf den Rechner beschränkt, von dem die HTML Seite geladen wurde

Beispiel/1

- Ein einfacher Zahlenaddierer

```
package adder;
import java.rmi.*;
public interface Adder extends Remote {
    public int add(int a, int b) throws RemoteException ;
}
```

Beispiel/2

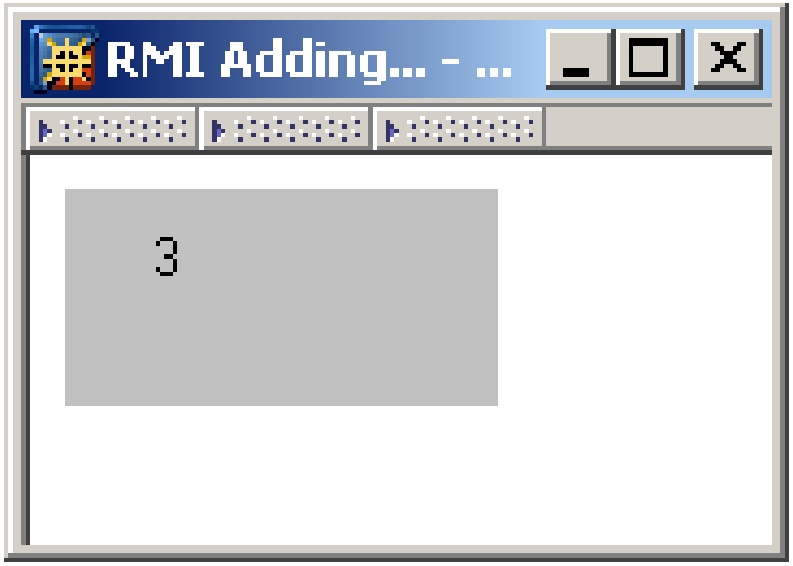
- Einfache Implementierung:

```
package adder;
import java.rmi.*;
import java.rmi.server.*;

public class AdderImpl extends UnicastRemoteObject implements
   Adder {

   public AdderImpl() throws RemoteException { super(); }
   public int add(int a, int b) throws RemoteException {return a+b;
   }

   public static void main(String[] argv) {
    try {
      AdderImpl ai=new AdderImpl();
      Naming.rebind("rmi://localhost/adder",ai);
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
   }
}
```

Beispiel/3

- HTML Seite

```
<head>
 <title>RMI Adding...</title>
</head>
<body>
 <applet codebase="/" code="adder.AdderApplet"
        width="100" height="50">
   <param name="a" value="1">
   <param name="b" value="2">
 </applet>
</body>
```



Beispiel/4 Das eigentliche Applet

Freie Universität Berlin

```
package adder;
import java.applet.Applet;
import java.awt.Graphics;
import java.rmi.*;
public class AdderApplet extends Applet {
    Adder adder=null;
    String result="";
    public void init() {
     try {
        adder=(Adder)Naming.lookup("//"+getCodeBase().getHost()+"/adder");
        int a=Integer.parseInt(getParameter("a"));
        int b=Integer.parseInt(getParameter("b"));
        result=Integer.toString(adder.add(a,b));
     } catch (Exception e) {
        System.out.println(e.getMessage());
     }
    }
    public void paint(Graphics g) {
        g.drawString(result,20,20);
    }
}
```

Zusammenfassung

Zusammenfassung

- Javascript
  - Einfache imperative Sprache
  - Eingebunden in Web-Seiten
  - Zugriff auf Dokumentenstruktur im Browser
  - Ereignisgesteuerte Ausführung
- AJAX
  - GUI Teile dynamisch verändern
  - Ermöglicht durch Asynchroner HTTP-Zugriff und Veränderung des Dokuments
- Einfache Applets
  - HTML Einbindung mit <applet>
  - Unterklasse von java.applet.Applet
  - Können auf Grafik-Display „schreiben“
  - init() und paint() Methoden
  - Interaktive Applets: Reaktion auf Ereignisse Applets und RMI
  - Wie „normales“ RMI
  - In der Regel auf Herkunftsserver beschränkt
- Applets mit Animation
  - run() für eigenen Thread
  - start() und stop() für Browserseitige Kontrolle

Literatur

- Stefan Münz: SelfHTML. http://www.selfhtml.org
- Sun Microsystems, Inc. The Java Tutorial. Trail: Writing Applets. http://java.sun.com/docs/books/tutorial/applet/index.html
- Sun Microsystems, Inc. Getting Started Using RMI. http://java.sun.com/j2se/1.4.2/docs/guide/rmi/getstart.doc.html
- AJAX:Getting Started
- Mozilla Developer Center. AJAX:Getting Started http://developer.mozilla.org/en/docs/AJAX:Getting_Started
- Mozilla Developer Center. AJAX. http://developer.mozilla.org/en/docs/AJAX

Anhang: Applets mit Animation

Applets mit Animation

- Applets implementieren die Schnittstelle Runnable des Pakets java.lang.
- Beispiel

```
public class MeinApplet
                extends Applet
                implements Runnable { ... }
```

- Die Methode public void run() ist die einzige zu implementierende abstrakte Methode der Schnittstelle Runnable.
  - Die Methode run() definiert, wie die Animation läuft.

Animationen in Applets (1)

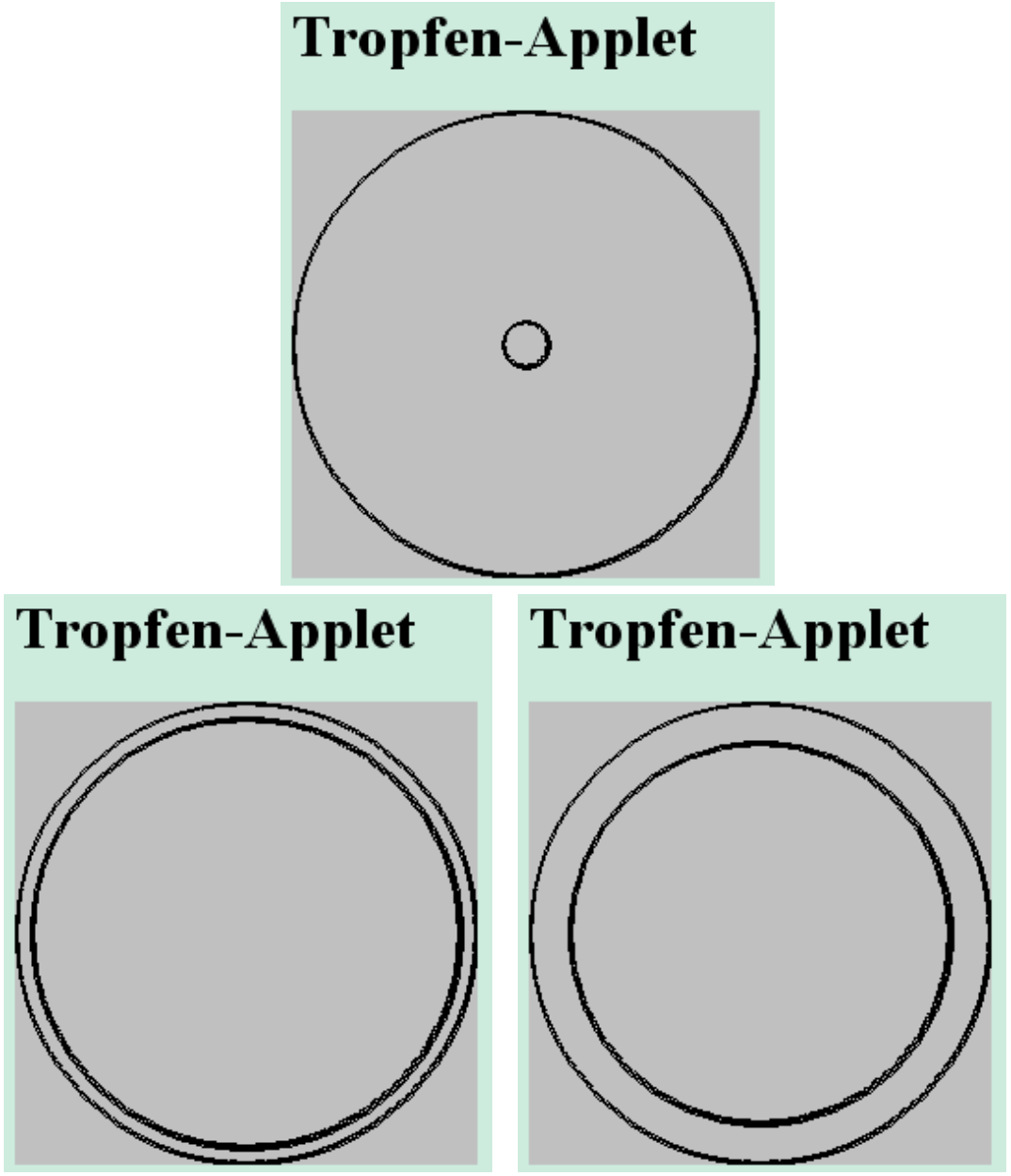
- Animationen laufen als Threads
  - Thread animationsthread = new Thread(applet);
- Zu definierende Methoden für Applets mit Animation:
  - public void start()
    - wird vom Browser zum Start des Applets nach der Methode init() aufgerufen und jedesmal, wenn es wieder im Browser angezeigt wird
    - muß den Animationsthread mit start() aus der Klasse Thread, welche die Methode run() aufruft, starten durch animationsthread.start();
    - überschreibt Methode aus java.applet.Applet

Animationen in Applets (2)

- Zu definierende Methoden für Applets mit Animation:
  - public void stop()
    - wird vom Browser zum Anhalten beim Verlassen der Seite aufgerufen
    - muß den Animationsthread mit der Methode stop() der Klasse Thread anhalten durch animationsthread.stop();
    - überschreibt Methode aus der Klasse java.applet.Applet

Beispiel: Applet mit Animation

```
<h1>Tropfen-Applet</h1>
<applet
   codebase="."
   code="Tropfen.class"
   width="250" height="250"
   alt="Das Topfen-Applet
        läuft leider nicht!">
<p>Hier sollte eigentlich ein
   <b>Topfen-Applet</b>
   dargestellt werden,
   aber ihr Browser
   versteht leider
   keine Applets.</p>
</applet>
```



Die Datei Tropfen.java (1)

Freie Universität Berlin

```
import java.applet.*; // fuer Applets
import java.awt.*;    // fuer Graphiken

public class Tropfen extends Applet implements Runnable {

  private Thread tropfen = null;

  public void start() {
    tropfen = new Thread(this);
    tropfen.start();
  } // Ende Methode start()

  public void run() {
    Graphics g = this.getGraphics(); // Graphik-Objekt
    Rectangle re = this.getBounds(); // Abmessungen des Applets
    int r=0,                 // aktueller Radius
        d,                   // aktueller Durchmesser
        breite=re.width/2, // halbe Applet-Breite
        hoehe=re.height/2; // halbe Applet-Hoehe
```

Die Datei Tropfen.java (2)

```
    while (true) { // Endlosschleife
      if (r>Math.max(breite,hoehe)) r = 0; d = 2*r;
      g.clearRect(0,0,re.width,re.height); // alten Kreis löschen
      g.drawOval(breite-r,hoehe-r,d,d);   // neuen Kreis zeichen
      g.drawOval(breite-(r-1),hoehe-(r-1),d-2,d-2); //Linienbr.2
      r++;
    } // Ende while
  } // Ende Methode run()
  public void stop() {
    tropfen.interrupt();
    tropfen = null;
  } // Ende Methode stop()
  public String[][] getParameterInfo() {
    String[][] info = {{}}; // kein Parameter
    return info;
  } // Ende Methode getParameterInfo()
  public String getAppletInfo() {
    return "Tropfen-Applet\nVon Wilhelm Weisweber";
  } // Ende Methode getAppletInfo()
} // Ende Klasse Tropfen
```

Applets mit Ressourcenbelegung

- Zu definierende Methoden für Applets, die Ressourcen belegen:
  - public void destroy()
    - wird vom WWW-Browser aufgerufen, wenn das Applet aus dem Browser entfernt wird nachdem die Methode stop() der Klasse Applet beendet ist
    - zur Freigabe alloziierter Ressourcen
    - überschreibt Methode aus der Klasse java.applet.Applet

Aufrufreihenfolge für Methoden

- (1) init()
- (2) paint(Graphics g)
- (3) start()
- (4) run()
- (5) stop()
- (6) destroy()
- alle Methoden sind public void