

JavaScript

- Einführung
- Einbindung in HTML
- Eventhandling
- Programmierung
 - Variablen, Operatoren, Funktionen
- DOM
 - Objekte
- Formularprüfung
- Google Maps



JavaScript?

- **Skriptsprache**, *clientseitige* Interpretation *im Browser*
- *Netscape / Navigator* , *Microsoft / Internet Explorer*
- *als* **ECMAScript** standardisiert, trotzdem als **JavaScript** bezeichnet
- früher: visuelle Spielereien oder Formularüberprüfung
- heute: *Web 2.0* / **AJAX** (interaktive Webanwendungen)



Clientseitiges JavaScript

- In HTML eingebettet
- Wird vom Browser ausgeführt (interpretiert)
- Zugriff auf das dargestellte *Dokument* (*Objekte des Browsers*)
- *Ereignisorientiert* (*Event-Handler driven*)
- Einbindung in HTML als **Skript**, über **Event-Handler** oder **URLs**



JavaScript & HTML

Das <script>-Tag

- Im Dokument als Inhalt zwischen dem <script>-Tag

```
<script type="text/javascript">  
    alert("Die Uhrzeit: " + new Date());  
</script>
```

- Als externe Datei unter Verwendung des **src-Attributs**

XHTML:

```
<script type="text/javascript" src="library.js" />
```

HTML 4:

```
<script type="text/javascript" src="library.js"></script>
```



JavaScript & HTML

<html>

<head>

</head>

<body>

Text vor der Meldung

<script type="text/javascript">

document.write("Hallo, Welt!");

</script>

Text hinter der Meldung

</body>

</html>

Text vor der Meldung
Hallo, Welt!
Text hinter der Meldung



JavaScript & HTML

JavaScript-URLs

- JavaScript-Code in einer URL mit dem Pseudoprotokoll `javascript:`
- Der Code wird ausgewertet und das Ergebnis in einen String umgewandelt
- Um das Dokument nicht mit dem Rückgabewert zu überschreiben den Operator `void` verwenden

```
<form action="javascript:formUebertragen()">
```

```
<a href="javascript:void pruefen()">jetzt pr&uuml;fen</a>
```

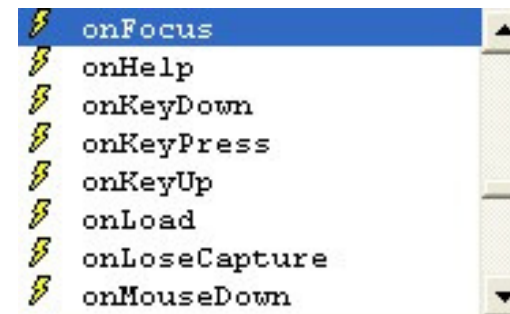
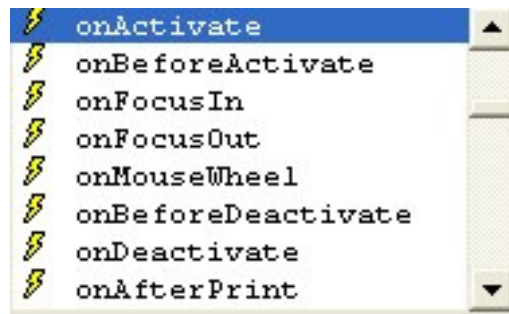


Event-Handler

- JavaScript als Wert für Event-Handler-Attribute von HTML-Tags
- Event-Handler beginnen mit „on“
- Der Code wird ausgeführt, sobald das Event eintritt

```
<input type = "button" value = "Klick mich! "  
      onClick = "alert('Hallo Welt!'); " />
```

Eventbeispiele



Event-Handler

onMouseOver()

Der Mauszeiger wurde über das Element bewegt.

onMouseOut()

Der Mauszeiger wurde wieder aus dem Element heraus bewegt.

onClick()

Der Anwender hat das Element angeklickt.

onLoad()

Die HTML-Seite wurde geladen.

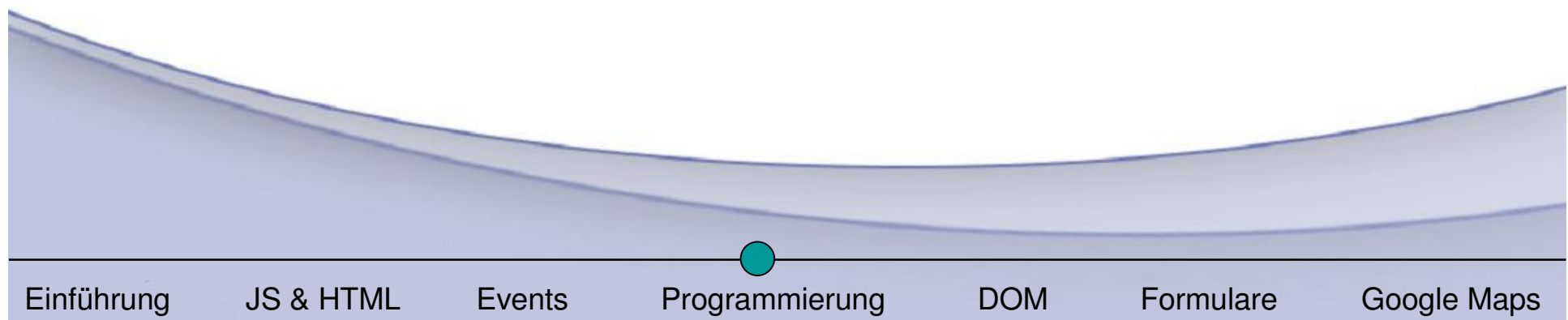
onUnload()

Die HTML-Seite wird gerade verlassen.

Programmierung

Grundlagen von JavaScript

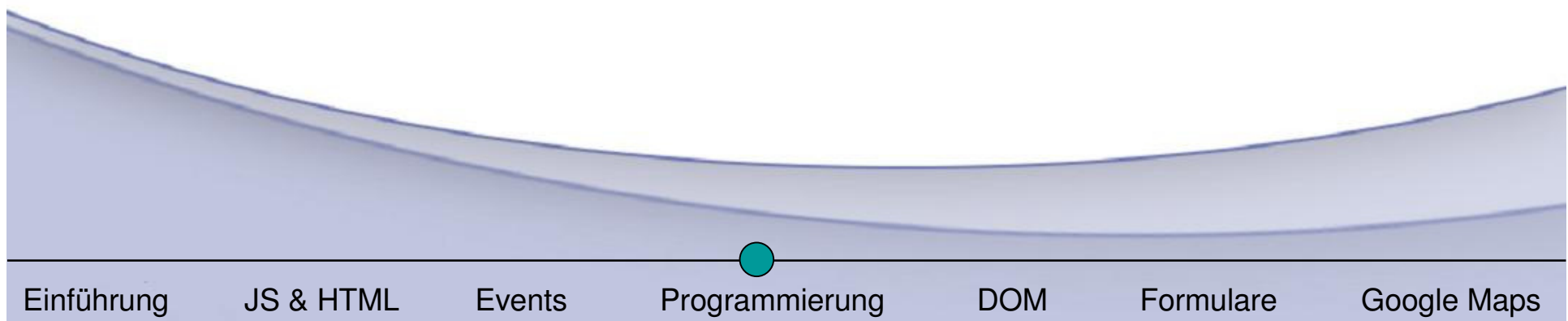
- stark **objektorientierte** *leichtgewichtige* Skriptsprache
- auch (ausschließlich) *funktional* einsetzbar
- *interpretiert*, nicht kompilierbar
- Event-Handler gesteuert
- I/O nur über Cookies und das DOM möglich



Programmierung

```
<script language="JavaScript"> <!--  
  var anzahl;  
  var name = "Schafe";  
  anzahl = 3;  
  var beine = 4*anzahl;  
  var satz = anzahl+" "+name+" haben "+beine+" Beine";  
  document.write(satz);  
--> </script>
```

- Kein Variablentyp nötig
- Arithmetische Operatoren
- Stringverkettung mit +
- Funktionsaufrufe (mit Parameter)



Programmierung

```
<script type="text/javascript">  
var d=new Date();  
var time = d.getHours();  
if (time<10) {  
    document.write("<b>Good morning</b>");  
}  
var i=0;  
for (i=0; i<=10; i++) {  
    document.write(„Nummer " + i);  
    document.write("<br />"); }  
</script>
```

Verzweigungen

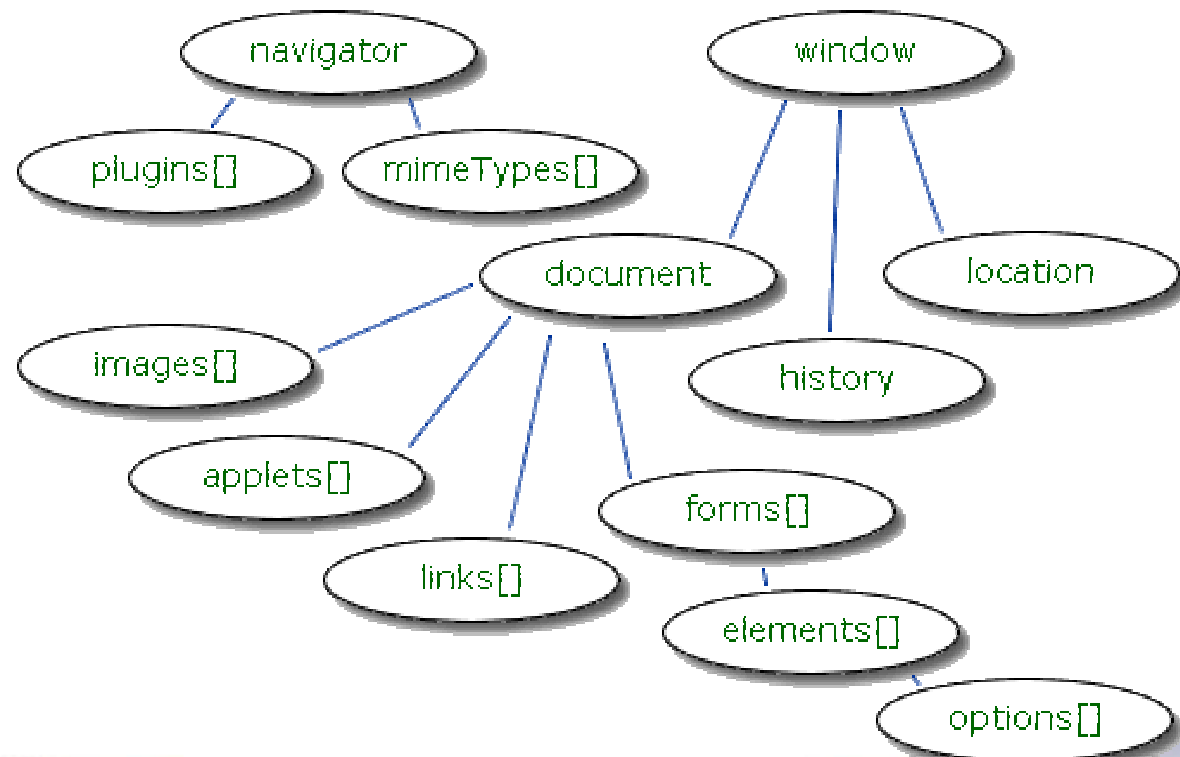
- Sag "Guten Morgen"
wenn es früher als 10 Uhr ist

Schleifen

- Zähle bis 10
- und gib dabei die Zahl aus

Document Object Model

- XML-Dokumente sind aufgebaut wie ein *Wurzelbaum*
- Jeder *Knoten*/jedes XHTML-Tag kann als ein **Objekt** angesprochen werden
- Das **document-Objekt** ist die *Wurzel*



Was bringt uns das?

Document Object Model

- Zugriff auf **Eigenschaften** und **Methoden** der Elemente
- Auslesen von Informationen, *Verändern* der *Dokumentstruktur* möglich
- *dynamische* Inhalte

```
<html>
<head>
<script type="text/javascript" src="library.js"></script>
</head>
<body>
  <form>
    <input type="button" value="Start" onClick="timedCount()">
    <input type="button" value="Stop" onClick="stopCount()">
    <input type="text" id="txt">
  </form>
  <div id="test" style="position:absolute; top:50px;">
    hallo
  </div>
</body>
</html>
```

```
<script type="text/javascript">
  var c=0;
  var t;
  function timedCount() {
    c = c+1;
    document.getElementById('txt').value = c;
    t = setTimeout("timedCount()",1000);
    var cont = document.getElementById("test");
    cont.style.left = c*10;
  }
  function stopCount(){
    clearTimeout(t);
  }
</script>
```

Beispiel zeigen

Einführung

JS & HTML

Events

Programmierung

DOM

Formulare

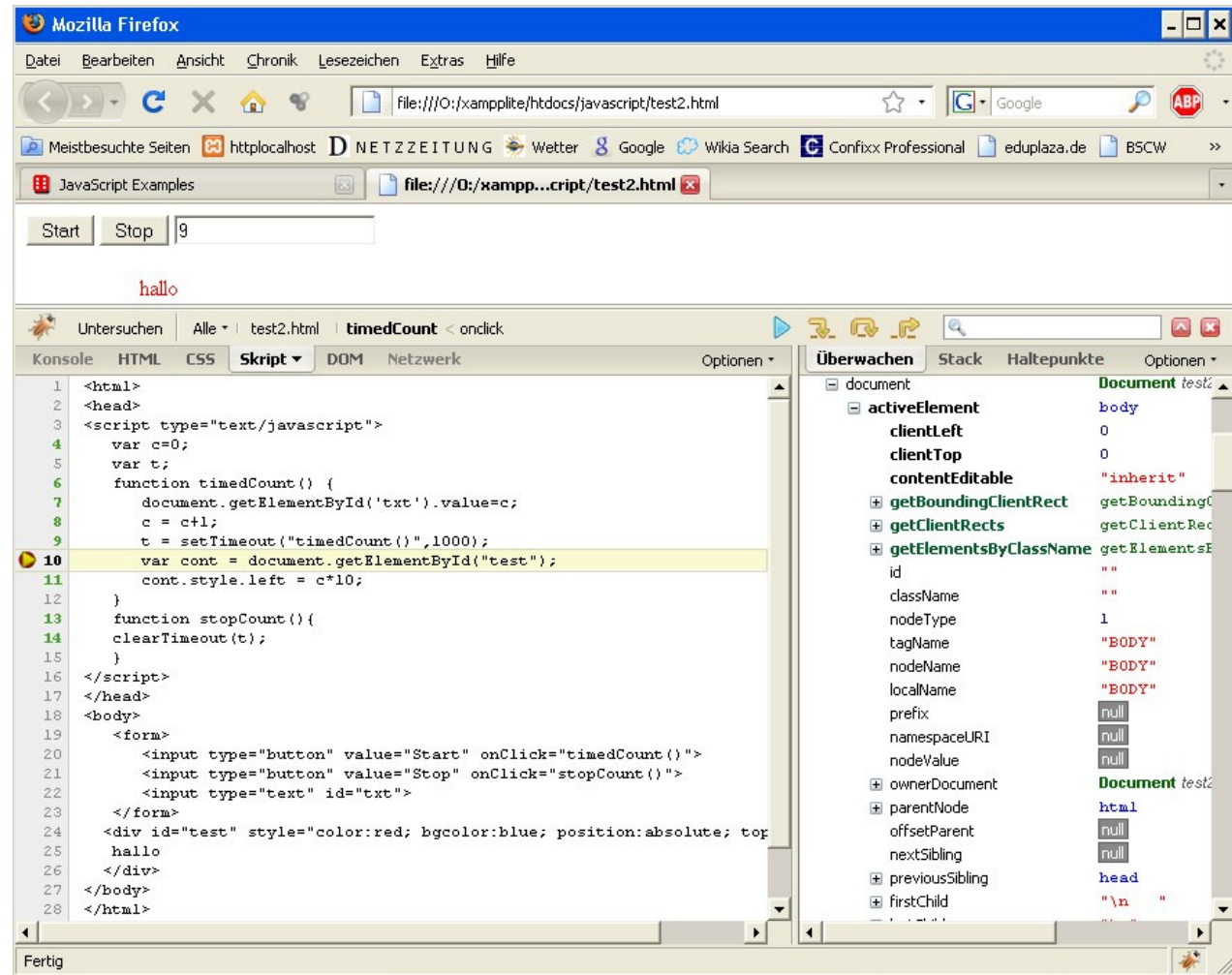
Google Maps

Document Object Model

FireBug für Mozilla

Debugging / Analyse fremder Dokumente

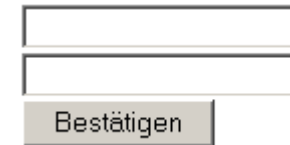
Muss manuell
installiert werden



Formular-Check

```
<form name="beispielform">
  <input type="text" name="vorname">
  <input type="text" name="nachname">
  <input type="button" onClick="checkform()">
</form>
```

```
function checkform() {
  if(document.beispielform.vorname.value == "") {
    alert("Sie haben keinen Vornamen eingegeben!");
  }
  if(document.beispielform.nachname.value == "") {
    alert("Sie haben keinen Nachnamen eingegeben!");
  }
}
```



A web form consisting of two empty text input fields stacked vertically, followed by a button labeled 'Bestätigen'.



Google Maps API

- Einbindung von Google Maps in die eigenen Webseite
- Bearbeiten der Karte mittels JavaScript
- Setzen von eigenen Markern



Quellenangaben

„JavaScript - kurz & gut“, *David Flanagan & Lars Schulten, O'Reilly, 2007*

„Webdesign mit JavaScript & Ajax“, *Heinle, Peña & Speidel, O'Reilly, 2006*

Die Google Maps API Dokumentation, <http://www.google.de/apis/maps>

<http://www.w3schools.com/JS/>



JavaScript

- Einführung
- Einbindung in HTML
- Eventhandling
- Programmierung
 - Variablen, Operatoren, Funktionen
- DOM
 - Objekte
- Formularprüfung
- Google Maps



JavaScript?

- **Skriptsprache**, *clientseitige* Interpretation *im Browser*
- *Netscape / Navigator* , *Microsoft / Internet Explorer*
- *als* **ECMAScript** standardisiert, trotzdem als **JavaScript** bezeichnet
- früher: visuelle Spielereien oder Formularüberprüfung
- heute: *Web 2.0* / **AJAX** (interaktive Webanwendungen)



Clientseitiges JavaScript

- In HTML eingebettet
- Wird vom Browser ausgeführt (interpretiert)
- Zugriff auf das dargestellte *Dokument* (*Objekte des Browsers*)
- *Ereignisorientiert* (*Event-Handler driven*)
- Einbindung in HTML als **Skript**, über **Event-Handler** oder **URLs**



JavaScript & HTML

Das <script>-Tag

- Im Dokument als Inhalt zwischen dem <script>-Tag

```
<script type="text/javascript">  
    alert("Die Uhrzeit: " + newDate());  
</script>
```

- Als externe Datei unter Verwendung des **src-Attributs**

XHTML:

```
<script type="text/javascript" src="library.js" />
```

HTML 4:

```
<script type="text/javascript" src="library.js"></script>
```



JavaScript & HTML

<html>

<head>

</head>

<body>

Text vor der Meldung

<script type="text/javascript">
document.write("Hallo, Welt!");

</script>

Text hinter der Meldung

</body>

</html>

Text vor der Meldung
Hallo, Welt!
Text hinter der Meldung



JavaScript & HTML

JavaScript-URLs

- JavaScript-Code in einer URL mit dem Pseudoprotokoll `javascript:`
- Der Code wird ausgewertet und das Ergebnis in einen String umgewandelt
- Um das Dokument nicht mit dem Rückgabewert zu überschreiben den Operator `void` verwenden

```
<form action="javascript:formUebertragen()">
```

```
<a href="javascript:void pruefen()">jetzt pr&uuml;fen</a>
```

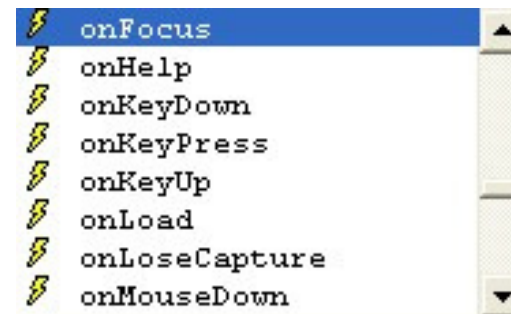
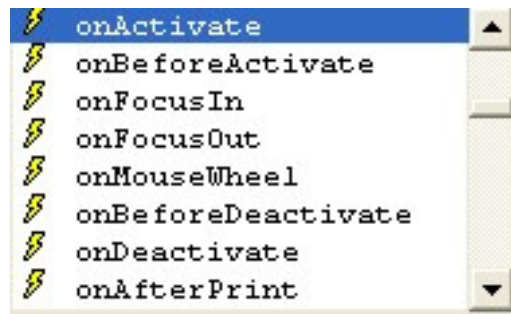


Event-Handler

- JavaScript als Wert für Event-Handler-Attribute von HTML-Tags
- Event-Handler beginnen mit „on“
- Der Code wird ausgeführt, sobald das Event eintritt

```
<input type = "button" value = "Klick mich! "  
      onClick = "alert('Hallo Welt!'); " />
```

Eventbeispiele



Event-Handler

onMouseOver()

Der Mauszeiger wurde über das Element bewegt.

onMouseOut()

Der Mauszeiger wurde wieder aus dem Element heraus bewegt.

onClick()

Der Anwender hat das Element angeklickt.

onLoad()

Die HTML-Seite wurde geladen.

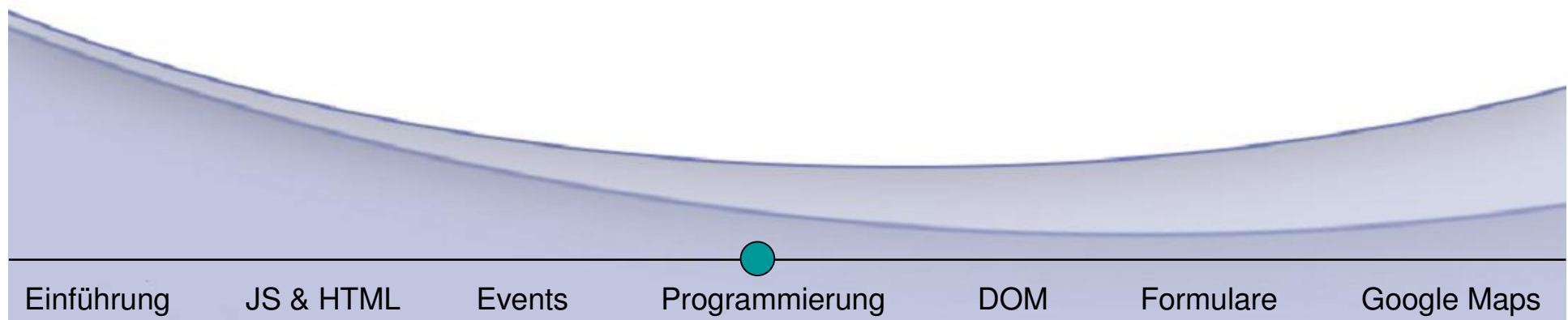
onUnload()

Die HTML-Seite wird gerade verlassen.

Programmierung

Grundlagen von JavaScript

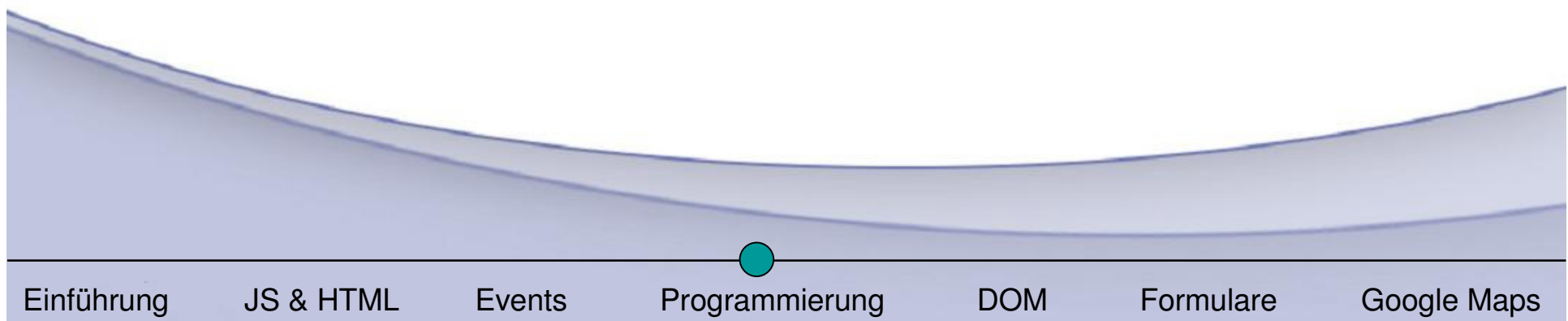
- stark **objektorientierte** *leichtgewichtige* Skriptsprache
- auch (ausschließlich) *funktional* einsetzbar
- *interpretiert*, nicht kompilierbar
- Event-Handler gesteuert
- I/O nur über Cookies und das DOM möglich



Programmierung

```
<script language="JavaScript"> <!--  
  var anzahl;  
  var name = "Schafe";  
  anzahl = 3;  
  var beine = 4*anzahl;  
  var satz = anzahl+" "+name+" haben "+beine+" Beine";  
  document.write(satz);  
--> </script>
```

- Kein Variablentyp nötig
- Arithmetische Operatoren
- Stringverkettung mit +
- Funktionsaufrufe (mit Parameter)



Programmierung

```
<script type="text/javascript">
var d=new Date();
var time = d.getHours();
if (time<10) {
    document.write("<b>Good morning</b>");
}
var i=0;
for (i=0; i<=10; i++) {
    document.write(„Nummer " + i);
    document.write("<br />"); }
</script>
```

Verzweigungen

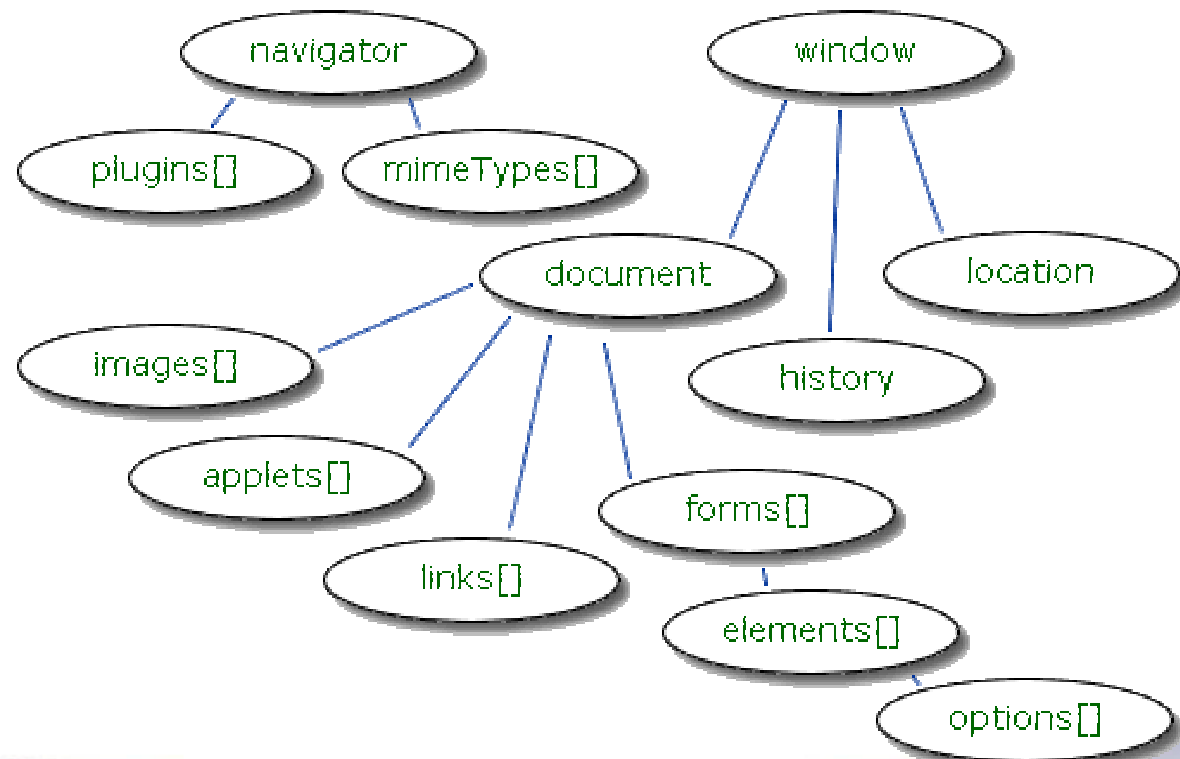
- Sag "Guten Morgen"
wenn es früher als 10 Uhr ist

Schleifen

- Zähle bis 10
- und gib dabei die Zahl aus

Document Object Model

- XML-Dokumente sind aufgebaut wie ein *Wurzelbaum*
- Jeder *Knoten*/jedes XHTML-Tag kann als ein **Objekt** angesprochen werden
- Das **document-Objekt** ist die *Wurzel*



Was bringt uns das?

Document Object Model

- Zugriff auf **Eigenschaften** und **Methoden** der Elemente
- Auslesen von Informationen, *Verändern* der *Dokumentstruktur* möglich
- *dynamische* Inhalte

```
<html>
<head>
<script type="text/javascript" src="library.js"></script>
</head>
<body>
  <form>
    <input type="button" value="Start" onClick="timedCount()">
    <input type="button" value="Stop" onClick="stopCount()">
    <input type="text" id="txt">
  </form>
  <div id="test" style="position:absolute; top:50px;">
    hallo
  </div>
</body>
</html>
```

```
<script type="text/javascript">
  var c=0;
  var t;
  function timedCount() {
    c = c+1;
    document.getElementById('txt').value = c;
    t = setTimeout("timedCount()",1000);
    var cont = document.getElementById("test");
    cont.style.left = c*10;
  }
  function stopCount(){
    clearTimeout(t);
  }
</script>
```

Beispiel zeigen

Einführung

JS & HTML

Events

Programmierung

DOM

Formulare

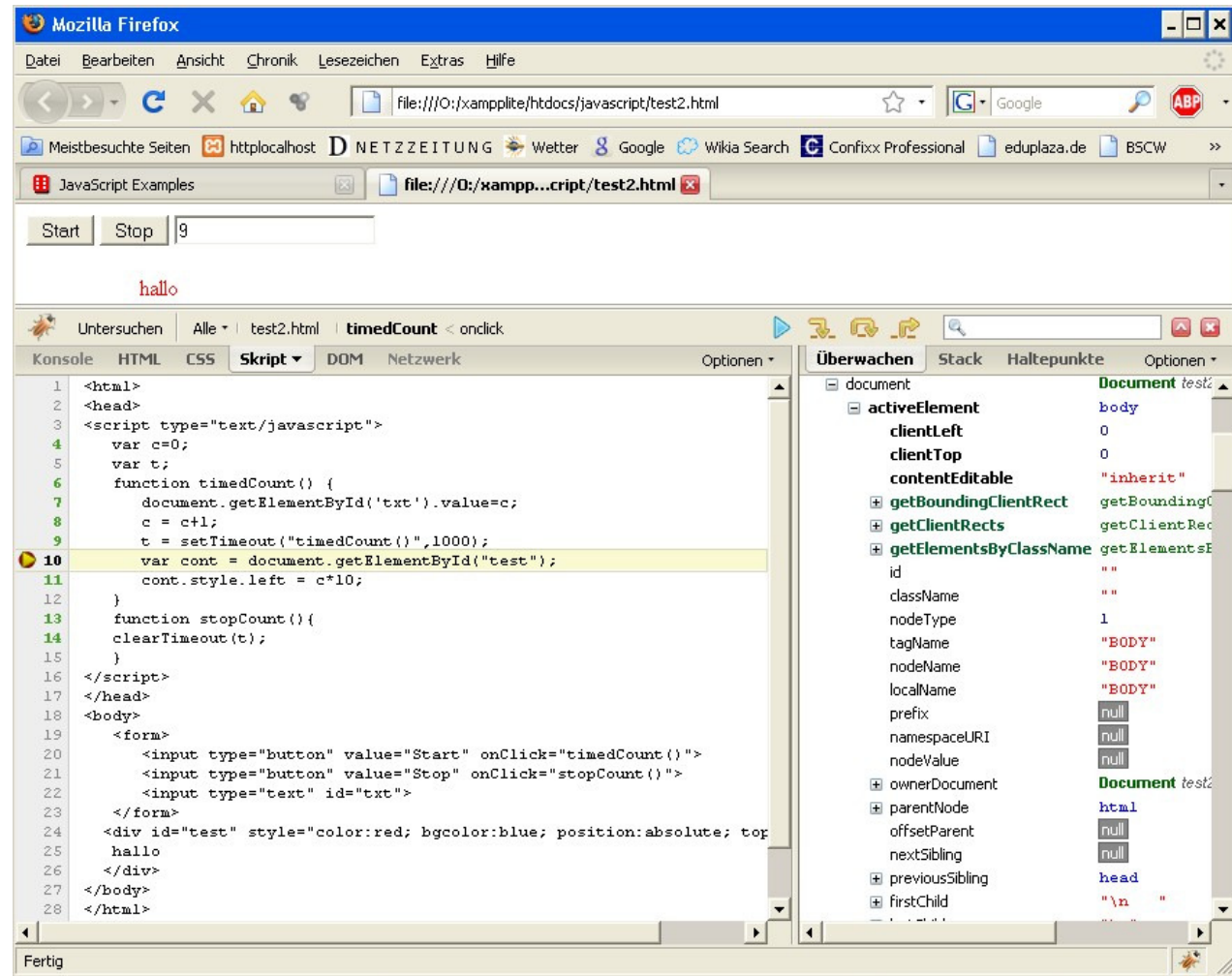
Google Maps

Document Object Model

FireBug für Mozilla

Debugging / Analyse
fremder Dokumente

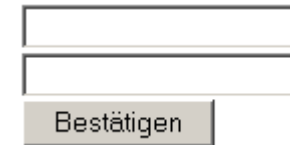
Muss manuell
installiert werden



Formular-Check

```
<form name="beispielform">  
  <input type="text" name="vorname">  
  <input type="text" name="nachname">  
  <input type="button" onClick="checkform()">  
</form>
```

```
function checkform() {  
  if(document.beispielform.vorname.value == "") {  
    alert("Sie haben keinen Vornamen eingegeben!");  
  }  
  if(document.beispielform.nachname.value == "") {  
    alert("Sie haben keinen Nachnamen eingegeben!");  
  }  
}
```



A web form with two empty text input fields stacked vertically. Below the inputs is a button labeled 'Bestätigen'.



Google Maps API

- Einbindung von Google Maps in die eigenen Webseite
- Bearbeiten der Karte mittels JavaScript
- Setzen von eigenen Markern



Quellenangaben

„JavaScript - kurz & gut“, *David Flanagan & Lars Schulten, O'Reilly, 2007*

„Webdesign mit JavaScript & Ajax“, *Heinle, Peña & Speidel, O'Reilly, 2006*

Die Google Maps API Dokumentation, <http://www.google.de/apis/maps>

<http://www.w3schools.com/JS/>



JavaScript

- Einführung
- Einbindung in HTML
- Eventhandling
- Programmierung
 - Variablen, Operatoren, Funktionen
- DOM
 - Objekte
- Formularprüfung
- Google Maps



JavaScript?

- **Skriptsprache**, *clientseitige* Interpretation *im Browser*
- *Netscape / Navigator* , *Microsoft / Internet Explorer*
- *als* **ECMAScript** standardisiert, trotzdem als **JavaScript** bezeichnet
- früher: visuelle Spielereien oder Formularüberprüfung
- heute: *Web 2.0* / **AJAX** (interaktive Webanwendungen)



Clientseitiges JavaScript

- In HTML eingebettet
- Wird vom Browser ausgeführt (interpretiert)
- Zugriff auf das dargestellte *Dokument* (*Objekte des Browsers*)
- *Ereignisorientiert* (*Event-Handler driven*)
- Einbindung in HTML als **Skript**, über **Event-Handler** oder **URLs**



JavaScript & HTML

Das <script>-Tag

- Im Dokument als Inhalt zwischen dem <script>-Tag

```
<script type="text/javascript">  
    alert("Die Uhrzeit: " + newDate());  
</script>
```

- Als externe Datei unter Verwendung des **src-Attributs**

XHTML:

```
<script type="text/javascript" src="library.js" />
```

HTML 4:

```
<script type="text/javascript" src="library.js"></script>
```



JavaScript & HTML

<html>

<head>

</head>

<body>

Text vor der Meldung

<script type="text/javascript">
document.write("Hallo, Welt!");

</script>

Text hinter der Meldung

</body>

</html>

Text vor der Meldung
Hallo, Welt!
Text hinter der Meldung



JavaScript & HTML

JavaScript-URLs

- JavaScript-Code in einer URL mit dem Pseudoprotokoll `javascript:`
- Der Code wird ausgewertet und das Ergebnis in einen String umgewandelt
- Um das Dokument nicht mit dem Rückgabewert zu überschreiben den Operator `void` verwenden

```
<form action="javascript:formUebertragen()">
```

```
<a href="javascript:void pruefen()">jetzt pr&uuml;fen</a>
```

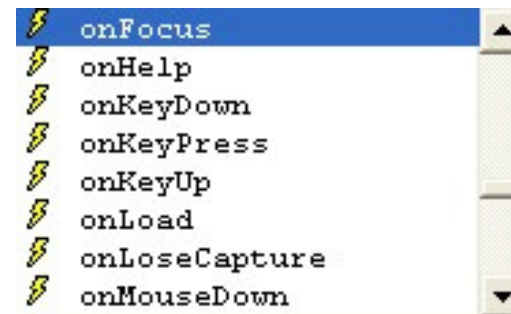
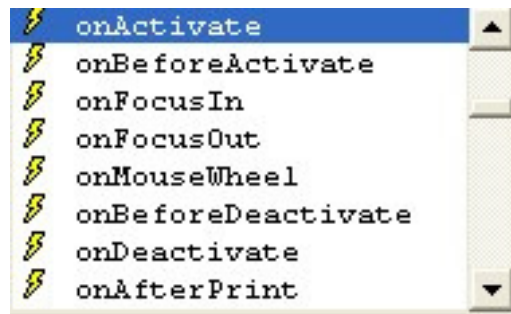


Event-Handler

- JavaScript als Wert für Event-Handler-Attribute von HTML-Tags
- Event-Handler beginnen mit „on“
- Der Code wird ausgeführt, sobald das Event eintritt

```
<input type = "button" value = "Klick mich! "  
      onClick = "alert('Hallo Welt!'); " />
```

Eventbeispiele



Event-Handler

onMouseOver()

Der Mauszeiger wurde über das Element bewegt.

onMouseOut()

Der Mauszeiger wurde wieder aus dem Element heraus bewegt.

onClick()

Der Anwender hat das Element angeklickt.

onLoad()

Die HTML-Seite wurde geladen.

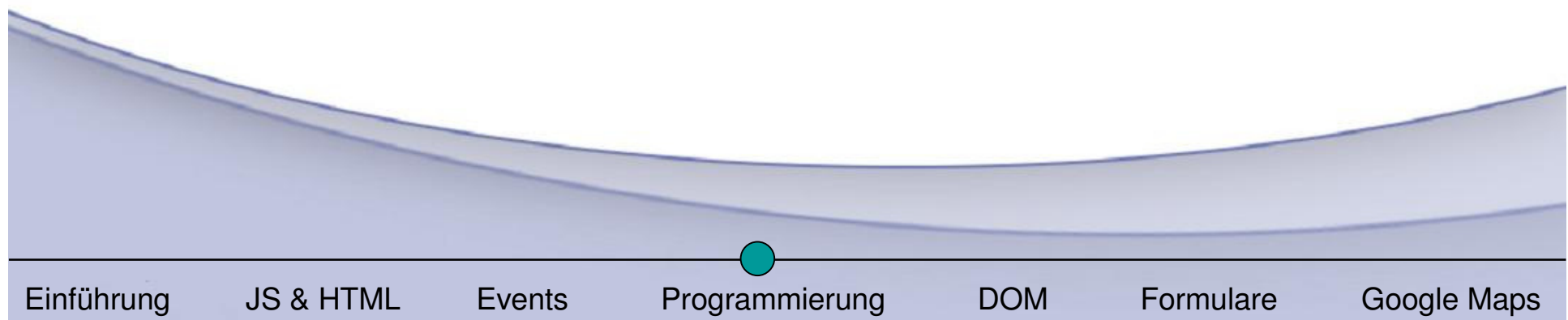
onUnload()

Die HTML-Seite wird gerade verlassen.

Programmierung

Grundlagen von JavaScript

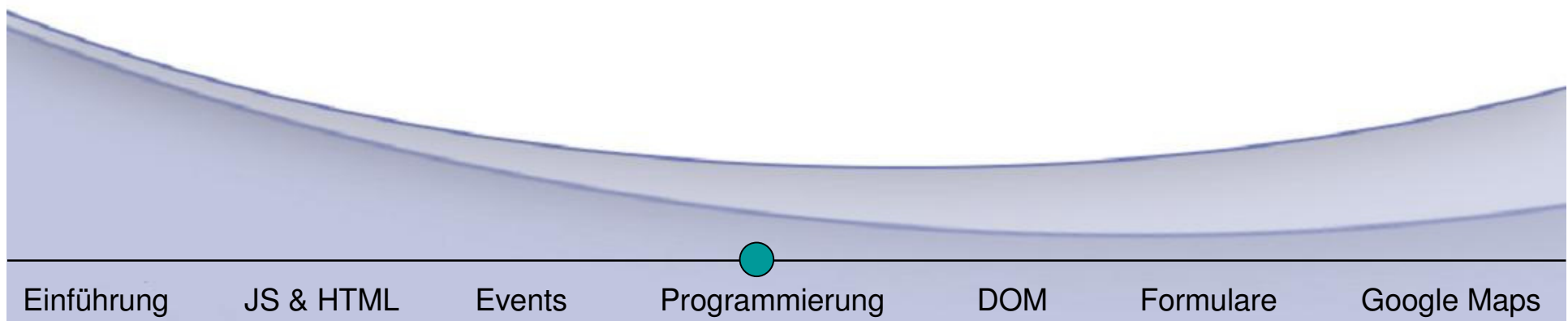
- stark **objektorientierte** *leichtgewichtige* Skriptsprache
- auch (ausschließlich) *funktional* einsetzbar
- *interpretiert*, nicht kompilierbar
- Event-Handler gesteuert
- I/O nur über Cookies und das DOM möglich



Programmierung

```
<script language="JavaScript"> <!--  
  var anzahl;  
  var name = "Schafe";  
  anzahl = 3;  
  var beine = 4*anzahl;  
  var satz = anzahl+" "+name+" haben "+beine+" Beine";  
  document.write(satz);  
--> </script>
```

- Kein Variablentyp nötig
- Arithmetische Operatoren
- Stringverkettung mit +
- Funktionsaufrufe (mit Parameter)



Programmierung

```
<script type="text/javascript">  
var d=new Date();  
var time = d.getHours();  
if (time<10) {  
    document.write("<b>Good morning</b>");  
}  
var i=0;  
for (i=0; i<=10; i++) {  
    document.write(„Nummer " + i);  
    document.write("<br />"); }  
</script>
```

Verzweigungen

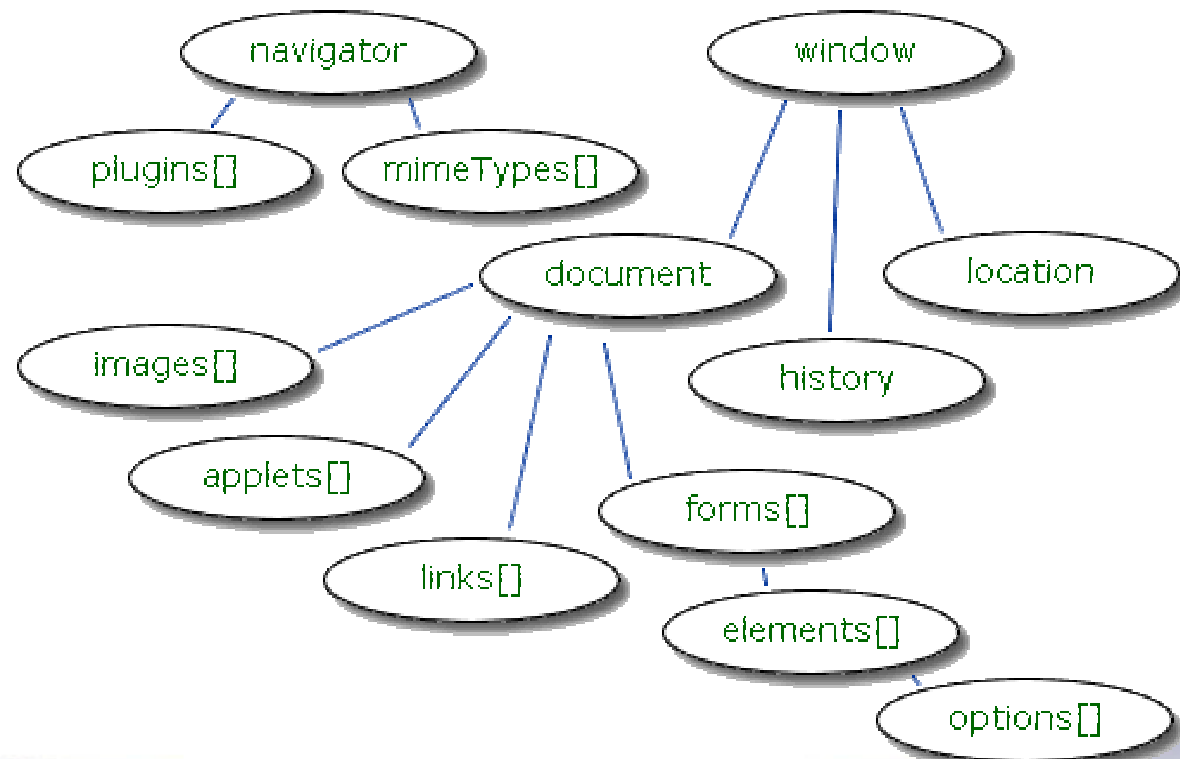
- Sag "Guten Morgen"
wenn es früher als 10 Uhr ist

Schleifen

- Zähle bis 10
- und gib dabei die Zahl aus

Document Object Model

- XML-Dokumente sind aufgebaut wie ein *Wurzelbaum*
- Jeder *Knoten*/jedes XHTML-Tag kann als ein **Objekt** angesprochen werden
- Das **document-Objekt** ist die *Wurzel*



Was bringt uns das?

Document Object Model

- Zugriff auf **Eigenschaften** und **Methoden** der Elemente
- Auslesen von Informationen, *Verändern* der *Dokumentstruktur* möglich
- *dynamische* Inhalte

```
<html>
<head>
<script type="text/javascript" src="library.js"></script>
</head>
<body>
  <form>
    <input type="button" value="Start" onClick="timedCount()">
    <input type="button" value="Stop" onClick="stopCount()">
    <input type="text" id="txt">
  </form>
  <div id="test" style="position:absolute; top:50px;">
    hallo
  </div>
</body>
</html>
```

```
<script type="text/javascript">
  var c=0;
  var t;
  function timedCount() {
    c = c+1;
    document.getElementById('txt').value = c;
    t = setTimeout("timedCount()",1000);
    var cont = document.getElementById("test");
    cont.style.left = c*10;
  }
  function stopCount(){
    clearTimeout(t);
  }
</script>
```

Beispiel zeigen

Einführung

JS & HTML

Events

Programmierung

DOM

Formulare

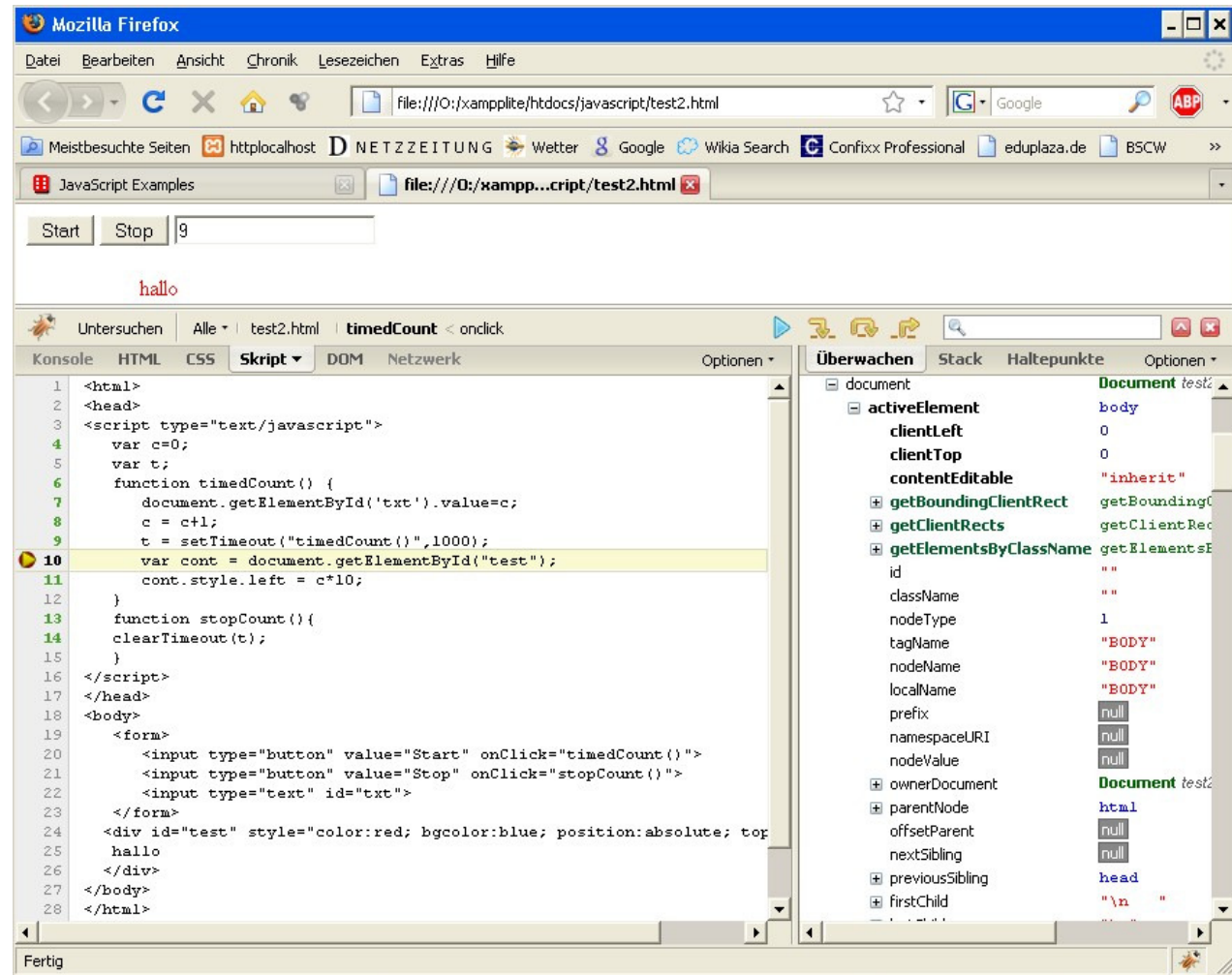
Google Maps

Document Object Model

FireBug für Mozilla

Debugging / Analyse
fremder Dokumente

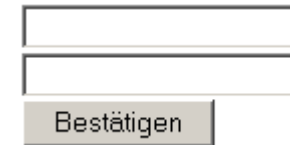
Muss manuell
installiert werden



Formular-Check

```
<form name="beispielform">  
  <input type="text" name="vorname">  
  <input type="text" name="nachname">  
  <input type="button" onClick="checkform()">  
</form>
```

```
function checkform() {  
  if(document.beispielform.vorname.value == "") {  
    alert("Sie haben keinen Vornamen eingegeben!");  
  }  
  if(document.beispielform.nachname.value == "") {  
    alert("Sie haben keinen Nachnamen eingegeben!");  
  }  
}
```



A web form with two text input fields stacked vertically. Below the fields is a button labeled 'Bestätigen'.



Google Maps API

- Einbindung von Google Maps in die eigenen Webseite
- Bearbeiten der Karte mittels JavaScript
- Setzen von eigenen Markern



Quellenangaben

„JavaScript - kurz & gut“, *David Flanagan & Lars Schulten, O'Reilly, 2007*

„Webdesign mit JavaScript & Ajax“, *Heinle, Peña & Speidel, O'Reilly, 2006*

Die Google Maps API Dokumentation, <http://www.google.de/apis/maps>

<http://www.w3schools.com/JS/>

