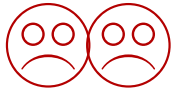
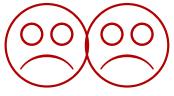
Programmierung von PDF-Dokumenten mit JavaScript im LaTeX-Quelltext

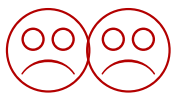
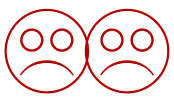
Ivica Rogina, Hochschule Karlsruhe

9. Mai 2012

Eigenschaften elektronischer Dokumentarten

	Portabilität	Interaktivität	Textsatzqualität
ASCII-Texte			
Word-Dateien			
HTML-Dokumente			
LaTeX→PDF-Dokumente		???	

Eigenschaften elektronischer Dokumentarten

	Portabilität	Interaktivität	Textsatzqualität
ASCII-Texte			
Word-Dateien			
HTML-Dokumente			
LaTeX→PDF-Dokumente			

Schöne und hässliche Dokumente

Verbreitete Unarten

- Missachtung der Orthographie
- Zusammenklauen von Illustrationen aus dem Web
- Missachtung von Textsatzregeln
- Wilder Mix der Schriftgrade, -familien und -stile
- Unharmonischer mathematischer Formelsatz

Vergessene Techniken

- Erst denken, dann tippen
- Kommuniziert ist nur was ankommt

Wie macht LaTeXDokumente „schön“?

- Textsatz mit Ligaturen und Kerning:
  fluffig vs. fluffig Word vs. Word
- ästhetische Leerräume
- ästhetische Verteilung der „Druckerschwärze“
- auch auf dem Papier genau so schön
- ästhetischer Formelsatz:

$$b_n = \frac{1}{\pi} \int_{x=0}^{T} f(x) \sin nx \, \mathrm{d}x \quad \text{vs.} \quad b_n = \frac{1}{\pi} \int_{x=0}^{T} f(x) \sin nx \, \mathrm{d}x$$

- und vieles weitere

Was kann der Adobe Reader bzw. Acrobat?

- **Darstellen** von Dokumenten im Portable Document Format (PDF)
- **Optimieren** der Darstellung z. B. durch:
  Antialiasing, Rendering, Skalierung, Vektorfonts, etc.
- Diverse mehr oder weniger nützliche **Funktionen** wie:
  Drucken, Suchen, Annotieren, cut/copy/paste, laut Vorlesen, etc.
- Darstellen und Ausfüllenlassen von **Formularfeldern**
- Interpretieren von **JavaScript** Quelltext.

Anwendungsmöglichkeiten für JavaScript in PDF

- Quellenangaben mit Link Versehen
- Prüfen, ob Formularfelder (korrekt) ausgefüllt sind
- Reaktion auf Inhalte der ausgefüllten Formularfelder
- Verwenden von Checkboxen und Radiobuttons
- Anzeigen von Meldungen und Animationen
- Durchführen komplexer Berechnungen

Beispiel: Farbcodeanzeige

R= G= B= $\Rightarrow$ Klick mich!

Man gebe Werte zwischen 0 und 255 für R,G,B ein und klicke auf die „Klick mich!“ oder die Farbbox.

Beispiel: Berechnungen durchführen

Das Skalarprodukt von $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ und $\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ ist

$x_1 \cdot y_1 + x_2 \cdot y_2 + x_3 \cdot y_3$

Hier zum selbst ausprobieren:

$$\begin{array}{lll} x_1 = & y_1 = & \\ x_2 = & y_2 = & \Rightarrow \vec{x} \cdot \vec{y} = \\ x_3 = & y_3 = & \end{array}$$

Hintern den Koeffizienten Werte eingeben und auf den Folgepfeil klicken.

Beispiel: Tests/Übungsklausuren

Wie viel ist 2+2?

Wie viel ist 2·3?

Andere Darstellung für 8-6:

Auswertung:

Und so geht's: Die insdljs Bibliothek

```
\begin{verbatim}
\documentclass{...}
...
\usepackage{insdljs}

\begin{insDLJS}[test]{test}{JavaScript}
    ...
\end{insDLJS}

\begin{document}
...
\end{document}
```

JavaScript Code

HTML-ähnlich Objekte
als LaTeX-Befehle
z. B. \PushButton

Buttons

Der LaTeX-Befehl

`\PushButton[onclick={` *JavaScript-Funktion* `}]{` *Beschriftung* `}`

erzeugt im PDF-Dokument eine klickbare Schaltfläche.

Beim Anklicken wird die angegebene Funktion aufgerufen.

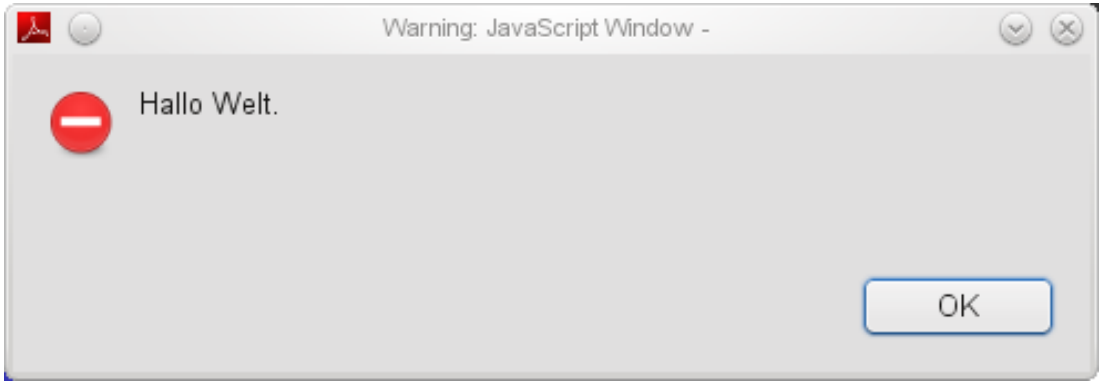
Diverse Optionen zum Erscheinungsbild existieren.

Alerts

Der JavaScript-Befehl

`app.alert(" `*Objekt*` ");`

lässt ein Pop-up-Dialog erscheinen mit dem zum String gewandelten Objekt.



Buttons und Alerts

```
\documentclass{...}
...
\usepackage{insdljs}

\begin{insDLJS}[test]{test}{JavaScript}
    function Hallo() { app.alert("Hallo."); }
\end{insDLJS}

\begin{document}
...
\PushButton[onclick={Hallo();}]{Klick mich!}
...
\end{document}
```

Klick mich!

Formularfelder

Der LaTeX-Befehl

`\TextField[name=` *Elementname* `]{` *Inhalt* `}`

erzeugt im PDF-Dokument ein ein editierbares Eingabefeld,

auf dessen Inhalt aus JavaScript mit

`this.getField("`*Elementname*`").value`

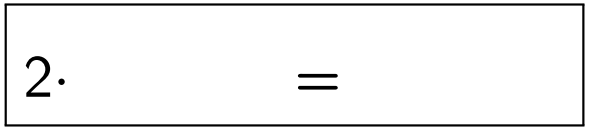
lesend u. schreibend zugegriffen werden kann.

Formularfelder

```
\documentclass{...}
...
\usepackage{insdljs}
\begin{insDLJS}[test]{test}{JavaScript}
    function Doppelt() {
      this.getField("Y").value=2*this.getField("X").value;
    }
\end{insDLJS}
\begin{document}
...
2$\cdot$\TextField[name=X,width=3em]{}
\PushButton[onclick={Doppelt();}]{=}
\TextField[name=Y,width=3em]{}
...
\end{document}
```

Formularfelder

```
\documentclass{...}
...
\usepackage{insdljs}
\begin{insDLJS}[test]{test}{JavaScript}
    function Doppelt() {
      this.getField("Y").value=2*this.getField("X").value;
    }
\end{insDLJS}
\begin{document}
...
2$\cdot$\TextField[name=X,width=3em]{}
\PushButton[onclick={Doppelt();}]{=}
\TextField[name=Y,width=3em]{}
...
\end{document}
```



Zahlreiche Erweiterungspakete

- **eforms**:
  komfortables Erstellen von Formularen
- **exerquiz**:
  Tests, Interaktive Prüfungen
- **anime**:
  Einbinden von Videos und Animationen inkl. Steuerelemente
- **acrotex**:
  alle o.g. und noch weitere

Schwierigkeiten

- Adobe Reader meldet gelegentlich Sicherheitswarnungen,
- Debugging ist (für Normalbenutzer) ziemlich umständlich,
- hohe Anfälligkeit für kleine Fehler
  (Sonderzeichen in Elementnamen, lange Elementnamen)

Zusammenfassung

- **portabel**:
  LaTeXund PDF sind auf mehr Systemen verfügbar als Word und HTML.
- **sieht gut aus**:
  LaTeXversteht Textsatzregeln (Word/HMTL nicht).
- **interaktiv**:
  fast die volle Mächtigkeit von JavaScript verwendbar.
- **spaßig**:
  Informatiker müssen es lieben.