

# WEB-TECHNOLOGIEN



Praktikum 11:  
DOM, Events, Node.js

# RAHMENAUFGABE: RAUMVERWALTUNG

- Im Rahmen des Praktikums entwickeln wir eine Web-Anwendung zur Verwaltung von Raumbuchungen (z.B. in einer Fachhochschule ;-)).
  - Diese Web-Anwendung werden wir Schritt für Schritt mit weiteren Anforderungen, Funktionen und Technologien erweitern.
- 
- ! **Verwenden Sie in den Laboren stets das Laufwerk "H:" zur Speicherung Ihrer Daten, damit Sie auf diese auch später (z.B. an einem anderen Laborrechner) noch zugreifen können.**
  - ! **Legen Sie Ihren Quellcode pro Praktikumsaufgabe in separaten Verzeichnissen (z.B. „praktikum2“, „praktikum3“) ab, damit die einzelnen Entwicklungsschritte bei den Abnahmen ersichtlich sind.**

# Praktikum 11: DOM, EVENTS, NODE.JS

Kopieren Sie den Stand von Praktikum 10 in ein neues Verzeichnis (z.B. „praktikum11“) und entwickeln Sie dort weiter.

## 1. Detailseite zu einem Raum per JavaScript erzeugen

In Praktikum 10 haben wir im externen JavaScript ein Raum- und ein Buchung-Objekt definiert und davon jeweils Instanzen erstellt. Insbesondere haben wir eine Raum-Instanz mit (mindestens) drei Buchung-Instanzen vorbereitet. Auf Basis dieser Raum-Instanz wollen wir nun die Inhalte der *Detailseite zu einem Raum* mit JavaScript erzeugen.

Entfernen Sie dazu auf der Seite die bisherigen statischen Beispieldaten aus dem HTML-Code. Erweitern Sie das externe JavaScript so, dass die Inhalte der Seite (d.h. die Informationen zum Raum sowie die Tabelle der Buchungen) mittels DOM-Manipulation erzeugt werden. Nutzen Sie die Möglichkeiten der DOM-API, um die benötigten Elemente im HTML-Code hinzuzufügen.

**Hinweis:** Konsultieren Sie die [Dokumentation zum Date-Objekt](#) für Informationen zum Lesen von Daten aus Date-Objekten!

# Praktikum 11: DOM, EVENTS, NODE.JS

## 2. Kacheln auf dem Dashboard hinzufügen

Wir wollen nun das Dashboard so erweitern, dass der Benutzer sich eigene Kacheln hinzufügen kann:

- a. „*Plus-Kachel*“: Erstellen Sie eine neue Kachel im Dashboard. Diese Kachel soll immer als letzte Kachel angezeigt werden und mit „+“ beschriftet sein. Die *Plus-Kachel* stellt dem Benutzer die Funktion zur Erstellung von neuen Kacheln zur Verfügung.
- b. Bei Klick auf die *Plus-Kachel* soll der Benutzer zur Eingabe eines Titels sowie einer URL für den Hyperlink der neu hinzuzufügenden Kachel aufgefordert werden.
- c. Hat der Benutzer diese Eingaben getätigt, so soll die neue Kachel per JavaScript erstellt und in das Dashboard eingefügt werden.

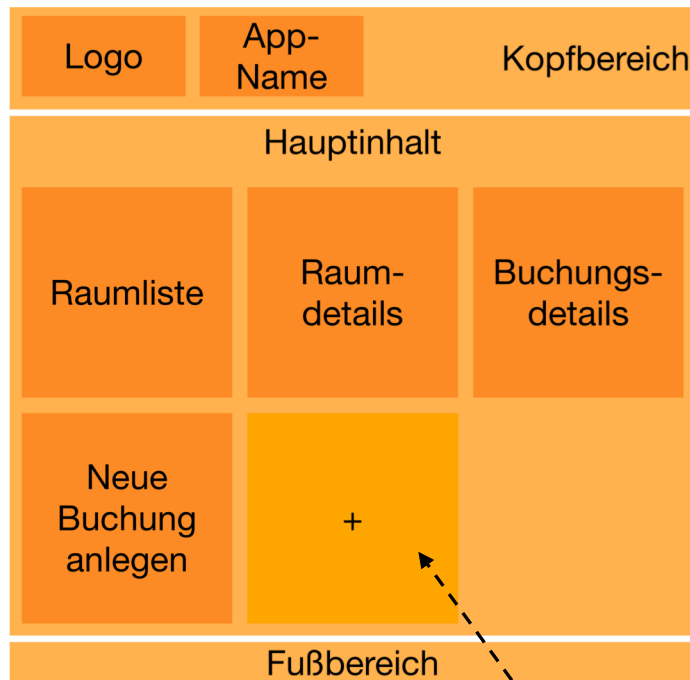
Verwenden Sie eine separate Datei `dashboard.js` für das externe JavaScript. Beachten Sie die Hinweise auf der folgenden Seite!

*Weiter auf der nächsten Seite...*

# Praktikum 11: DOM, EVENTS, NODE.JS

## 2. Kacheln auf dem Dashboard hinzufügen (Fortsetzung)

### Skizze zum Dashboard:

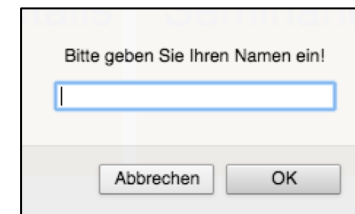


„Plus-Kachel“

### Hinweis:

Das `window`-Objekt bietet die Methode `prompt(message)` zum Anzeigen eines Dialoges mit Texteingabefeld. Beispiel:

```
/* Rückgabewert von prompt ist der vom Benutzer  
   eingegebene Wert */  
let name  
  = prompt("Bitte geben Sie Ihren Namen ein!");
```



# Praktikum 11: DOM, EVENTS, NODE.JS

## 3. Node.js einrichten und ausprobieren

In dieser Aufgabe bereiten wir Praktikum 12 vor. Ziel der Aufgabe ist es, eine lauffähige Node.js-Umgebung zu haben.

1. Auf den Rechnern im Labor ist Node.js bereits vorinstalliert. Falls Sie also mit einem Laborrechner arbeiten, so können Sie direkt mit Schritt 4 fortfahren.
2. Falls Sie mit einem eigenen Rechner arbeiten, so laden Sie den für Ihr Betriebssystem passenden Node.js-Installer hier herunter:  
<https://nodejs.org/de/download/>

### Downloads

Aktuellste LTS-Version: **10.14.2** (includes npm 6.4.1)

Lade den Node.js-Quellcode oder ein bestehendes Installationsprogramm für deine Plattform herunter und beginne gleich mit der Entwicklung.



**! Bitte wählen Sie unbedingt die LTS-Variante statt der aktuellsten (*current*) Version, um Stabilitätsprobleme zu vermeiden!**

# Praktikum 11: DOM, EVENTS, NODE.JS

## 3. Node.js einrichten und ausprobieren (Fortsetzung)

### 3. Testen Sie Ihre Node.js-Installation:


- a. Öffnen Sie eine Konsole, z.B.:
    - *Eingabeaufforderung* auf Windows 10: Tastenkombination **Windows + R** drücken, im sich öffnenden Fenster „cmd“ eingeben und Taste **Enter** drücken
    - *Terminal* auf MacOS: Tastenkombination **cmd + Leertaste** drücken, im sich öffnenden Fenster „terminal“ eingeben und Taste **Enter** drücken
  - b. Geben Sie in der Konsole den Befehl `node -v` ein. Falls Node.js korrekt installiert ist, sollte nun die Node-Version angezeigt werden (z.B. „v10.14.2“).
- ### 4. Speichern Sie das Programm aus Aufgabe 2 (CODING KATA) von Übungsblatt 10 in eine Datei namens `tictac.js` in Ihr „praktikum11“-Verzeichnis (lösen Sie die Aufgabe ggf. zuerst, falls Sie die entsprechende Übung nicht besucht haben).

# Praktikum 11: DOM, EVENTS, NODE.JS

## 3. Node.js einrichten und ausprobieren (Fortsetzung)

5. Wechseln Sie nun auf der Konsole (siehe Schritt 3a) in Ihr „praktikum11“-Verzeichnis (Befehl `cd <Pfad zum Verzeichnis>`) und führen Sie dort den folgenden Befehl aus: `node tictac.js`. Unser Programm sollte ausgeführt werden und eine Ausgabe wie diese produzieren:

```
sven@tsu:~/Documents/lehre/webtechnologien/veranstaltung/praktikum/praktikum11 (wise1819) $ node tictac.js
1 2 Tic 4 Tac Tic 7 8 Tic Tac 11 Tic 13 14 TicTac 16 17 Tic 19 Tac Tic 22 23 Tic Tac 26 Tic 28 29 TicTac 31 32 Tic 34 Tac Tic 37 3
8 Tic Tac 41 Tic 43 44 TicTac 46 47 Tic 49 Tac Tic 52 53 Tic Tac 56 Tic 58 59 TicTac 61 62 Tic 64 Tac Tic 67 68 Tic Tac 71 Tic 73
74 TicTac 76 77 Tic 79 Tac Tic 82 83 Tic Tac 86 Tic 88 89 TicTac 91 92 Tic 94 Tac Tic 97 98 Tic Tac
```

-  **Tipp:** In Visual Studio Code können Sie über einen Rechtsklick auf die Datei und Auswahl des Menüpunktes *Open in Terminal* eine integrierte Konsole öffnen.