1. Übungsblatt WebP2

2.1

User anlegen

* Zuerst wird eine Klasse definiert mit allen Eigenschaften des Nutzers
* Dann folgt eine Funktion mit dem Namen addUser mit einem Formelement und einem Inputelement für die jeweiligen Eigenschaften der Klasse
* Der Nutzer gibt auf der Webseite seine Eingaben ein und der Server liest diese erstmal und danach überprüft er die Daten
* Diese Überprüfung findet mit Hilfe einer if-else-Bedingung statt: wenn alle Werte eingetragen wurden vom Nutzer, dann erstellt der Server einen neuen User und dieser wird dann zur Userlist hinzugefügt durch userlist.push wenn allerdings nicht alle Inputfelder ausgefüllt sind, wird kein neuer User hinzugefügt. Stattdessen kommt eine Nachricht, die darauf hinweist, dass die Felder nicht vollständig ausgefüllt sind

2.2

User editieren

* Der User kann auf seine Daten zugreifen und diese verändern, dass funktioniert in dem er auf den Button mit dem Stift klickt
* Durch die const-html-input-Elemente kann der Nutzer die Eingaben verändern
* Sobald die Eingaben der Eigenschaften verändert wurden, liest der Server sie zuerst
* Danach überprüft er, ob die Eingaben eingetragen wurden
* Dies funktioniert mit einer for-Schleife, die auf den vorher angelegten User der Userlist zugreift
* Es folgt wieder eine if-else Bedingung, die genau wie beim User anlegen überprüft, ob alle Werte eingetragen wurden, falls nicht, wird der User nicht editiert

2.3

User Löschen

* Der User kann seine Eingaben löschen, indem er auf den Button mit der Mülltonne klickt
* Durch die Funktion deleteUser wird auf den User zugegriffen
* Event.target ist typisch, um den User zu löschen

3.1

Zeile 70 html:

<label for=”suche”> Benutzer suchen </label>

<input type=“search“ id=“Benutzersuche“ placeholder=“Benutzer suchen“>

<form action=“#“>

<p>

<lable>

Userlist <input type=“search“>

List=“User“

<lable>

<p>

3.2

Class Animal {

Public animalname: string;

Public animalyear: number;

Public animalgender: string;

Constructor(animalname: string, animalyear: number, animalgender: string) {

This.animalname = Name;

This.animalyear = Year;

This.animalgender = Gender;

}

}

3.3

Class User extends Animal {

Public AnimalName: string;

Public AnimalYear: number;

Public AnimalGender: string;

Constructor( firstname: string, lastname: string, creationDate: Date, animalname: string, animalyear: number, animalgender: string)

Super (firstname,lastname,creationDate);

This.animalname = string;

This.animalyear = number;

This.animalgender = string;

}

}

b)

function addUser(event)

const animalField: HtmlInputElement = document.getElementByclass (elementclass: ‘add-animal-input’) as HTMLInputElement;

const animal: string = animalField.value;

if (firstname && lastname && animal)

4.1

4.2

4.3

4.4

Get-Request:

* Ein Get Parameter wird verwendet, um Daten vom Server zu holen, wobei nur wenige Daten mitgeschickt werden können
* Daten werden in Kopfzeilen des Requests übertragen und sind in der URL Zeile des Browsers sichtbar, zudem können sie nur leicht manipuliert werden

Post-Request:

* Ein Post Parameter wird verwendet, um Daten zum Server zu senden und dort verarbeiten zu lassen
* Daten werden nicht nicht im Kopf, sondern als Nutzdaten übertragen und sind nicht sichtbar, zudem können sie nicht manipuliert werden