|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ECMA Script 6 (Primitves: Copy und Call by Value / Objects: Copy und Call by Reference)** | | | | | | | | | | | | | | | |
| **call()** und **apply()** sind vordefiniert Funktionen, die als 1 Arg. das Objekt entgegen nehmen und darauf die Funktion ausführen. | | | | | | | | | | | | | | | |
| **Default Value** (nicht übergebene Params = undefined) | | | **let** from = parseInt(query.von) || 0; *function*(text = **"defaultVal"**){ ... };  **function({** a **=** "x"**,** b **=** "y"**}** **=** **{})** **({** b **=** "1" **});** | | | | | | | | | | | | |
| **Variable Deklaration** var → undefined let → ReferenceError | | | Temporal Dead Zone: Da **kein Hoisting**, Variable in TDZ, wenn vor Deklaration genutzt wird. | | | | | | | | | | | | |
| **let zorn="hell"; const** number = 7; | | Variable, nur in umgebenem Scope, bei Hoisting nicht änderbar:  Konstante; nicht änderbar | | | | | | | | | | |
| **Objekte** | | | **const** counter = { **count** : 0}; | | | | | | { **car**, [**zorn**] : 2 } == { car: 1, hell: 2} | | | | | | |
| **Spread-Operator (…)** | | | **function** *myFunc*(...numbers){ numbers == [x,y,z,...]}; **const** numArray = [1,2,3]; **function** *myFunc*(...numArray); *// == myFunc(1,2,3);* | | | | | | | | | | | | |
| **Lambdas** | | | **array**.forEach((val) => {alert(val);}); *// this → context* | | | | | | | | | | | | |
| **String Template** | | | **`Content of** ${myVar}**`** *// use backticks!* | | | | | | | | | | | | |
|  | | | | | | | | | | | | | **function** *braches*(word){  **return** word  .toLowerCase()  .split(**''**)  .map( **function** (a, i, w) {  **return** w.indexOf(a) ==  w.lastIndexOf(a)?**'('**:**')'** }).join(**''**);} | | |
| **NodeJS (window, document sind nur im Browser vorhanden !‼) (console object ist auch in Node verfügbar)** | | | | | | | | | | | | | | | |
| **let server**=http.createServer(**function**(req, res){ *//Streams*  **let stream**=fs.createReadStream(\_\_dirname+ **'/data.txt'**);  *stream*.pipe(res); }); | | | | | | | | | | | | global=global||**window**; global 🡪 node window 🡪browser | | | |
| **Node:** Ein Thread, delegiert für jeden Event einen Subthread ans OS. (Kein Multithreading Overhead, Achtung Blocking Calls) | | | | | | | | | | | | | | | |
| **Express (Event Driven Node Server) / MVC Beispiel** | | | | | | | | | | | | | | | |
| **Main Application (/app.js) → Middleware registrieren (ChainOfResponsibility) 🡪 Server starten** | | | | | | | | | | | | | | | |
| **let express** = *require*(**'express'**); **var** path = *require*(**'path'**); **let** *session* = require(**'express-session'**); **let bodyParser** = *require*(**'body-parser'**); **let** app = **express**(); *// view setup:*  **let hbs** = require(**'hbs'**); **hbs**.registerHelper(**'rating'**, require(**'./helper/rating'**));  app.set(**'view engine'**, **'hbs'**);app.set(**'views'**, path.join(\_\_dirname, **'views'**)); app.set(**'port'**, port); app.set(**'hostname'**, **'localhost'**) ; *// register new middleware, order is important* app.use(**bodyParser**.**json**()); app.use(**bodyParser**.**urlencoded**({ **extended**: **false** })); app.use(*session*({ **secret**: **'test'**, **sortorder**: **"title"**})); app.use(*require*(**'less-middleware'**)(path.join(\_\_dirname, **'public'**))); app.use(**express**.**static**(path.join(\_\_dirname, **'public'**)));app.use(**function**(err, req, res, next){ *//errorhandler → last middleware*  ***console***.error(err.**stack**);  res.**status**(500).send(**`Error occured:** ${err.stack}**`**); });  **let** index = *require*(**'./routes/index'**); app.use(**'/'**, index); *//← routes* *// start server* **let** http = *require*(**'http'**); **let** server = http.createServer(app); app.listen(app.get(**'port'**), **function** () {  ***console***.log(**`Server is running at** http://${app.get('hostname')}:${ app.get(**'port'**)}**/`** )}); module.exports = app; | | | | | | | | | | | | | | | |
| **Routen (/routes)** | | | | | | | | | | | | | | | |
| **let express** = *require*(**'express'**); **let** router = **express**.Router(); **let** controller = *require*(**'../controller/indexController'**); **let** util = *require(***'../util/security'**); *// routes:* router.all(**'/\*'**, util.handleAuthenticate); *// via session cookie*  router.**get**(**'/'**, controller.showIndex()); router.**post**(**'/order/new'**, controller.createOrder());  router.**get**(**'/order/:id'**, controller.getOrder()); *// req.params.id* router.**post**(**'/order/:id'**, controller.setOrder());*// req.params.id* app.route(**'/multipleActions'**).get(**function**(req, res){..}).post(**function**(req, res) {..}) module.exports = router; | | | | | | | | | | | | | | | |
| **Controller (/controller)** | | | | | | | | | | | | | | | |
| **let** model = *require*(**"../services/orderService.js"**);  module.exports.showIndex = **function** (req, res) { **let** orders = **model**.getOrders(/\*callback\*/);  **switch** (req.**session**.**sortorder**) { **case "title"**: orders.sort(**function** (order1, order2) {  **return** order1.**title** < order2.**title**; }); **break**;  } res.render(**'index'**, {**orders**: orders});};  module.exports.getOrder = **function** (req, res, next) {  **let** orderId = req.**params**.**id**; //siehe routes (?id=2&name=Test -> req.query.\*)  model.getOrder(orderId, **function** (err, data) {  **if** (err) { ***console***.log(**"Database error: "**, err); next(err); }  res.render(**'order-edit'**, {**order**: data, **config**: config}); });}; module.exports.createOrder = **function** (req, res, next) {  **let** newOrder = { **title**: req.body.**title**, **dueDate**: req.body.dueDate };  model.createOrder(newOrder, **function** (err, newOrder) {  **if** (err) { ***console***.log(**"Database error: "**, err); next(err); }  res.redirect(302, **'/'**); }); }; | | | | | | | | | | | | | | | |
| **Model (/services)** | | | | | | | | | | | | | | | |
| **let** Datastore = *require*(**'nedb'**); **let** db = **new** Datastore(  {**filename**: **'data/orders.db'**, **autoload**: **true**, **timestampData**: **true**});  **function** Order(orderData){this.title = orderData['title']; ... }  module.exports.createOrder = **function** *createOrder*(orderData, callback) {  **let** newOrder = **new** Order(orderData);  db.insert(newOrder, **function**(err, newOrder) {  **if** (callback) { callback(err, newOrder); } }); } module.exports.getOrder = **function** *getOrder*(id, callback) {  db.findOne({**\_id**: id}, **function**(err, doc){ callback(err, data); }); }  module.exports.getOrders = **function** *getOrders*(callback) {  db.find({}, **function**(err, doc){ callback(err, data); });  }  module.exports.deleteOrder = **function** *deleteOrder*(id, req, res) {  db.delete({id}, **function**(err, doc){ callback(err, doc); });  } module.exports.updateOrder = **function** updateOrder(id, orderData, callback) {  db.update({**\_id**: id}, orderData, **function**(err, doc) { callback(err, doc); }); } | | | | | | | | | | | | | | | |
| **View (/views) (.hbs)** | | | | | | | | | | | | | | | |
| <**form**> {{#**with order**}}  <**input type="date" name="dueDate" placeholder="DD/MM/YYYY"   value="**{{**dueDate**}}**" required autofocus** />  <**button formmethod="POST" formaction="./save"**>Save</**button**>  <**button formmethod="GET" formaction="./"**>Reset</**button**>  {{/**with**}} </**form**> | | | | | | | | <**section class="content"**>  {{#**if orders**}}{{#**each orders**}}   <**article**><**p**>{{**dueDate**}}</**p**></**article**>  <**a href="./e/**{{**id**}}**"**>edit</**a**>{{/**each**}}  {{**else**}}<**p**>no orders</**p**>  {{/**if**}} </**section**> | | | | | | | |
| **Session / Cookies / Token** | | | | | | | | | | | | | | | |
| **Cookies** | | Client Cookie (benötigt Cookie Middleware) | | | | | | | | | | | | | |
| **Session** | | SessionID auf Server und als Cookie beim Client (Wiederspricht REST) | | | | | | | | | | | | | |
| **Token** | | Vom Server generiertes Token. Bei jeder Anfrage wird das Token übertragen. | | | | | | | | | | | | | |
| **REST: Representational State Transfer (Für klare Trennung zwischen Client und Server) → Best Practice für skalierbare Services** | | | | | | | | | | | | | | | |
| 1. Der Server interessiert sich nicht für Client Arbeit und umgekehrt (Strikte Trennung) 2. Statuslose Kommunikation. Jeder Request beinhaltet benötigte Infos für den Server (Erlaubt hohe Skalierung) 🡪 Session durch Token ersetzen 3. Clients können Antworten cachen (sofern erlaubt) 4. Layered: Erlaubt Einsatz von Loadbalancer, Proxies, Firewalls 5. Eine REST Schnittstelle sollte selbstbeschreibend sein. Alle Ressourcen sind identifziert über URI, Content Type, etc. | | | | | | | | | | | | | | | |
| **ROA: Resource Oriented Architecture 🡪 Ressource steht im Mittelpunkt** | | | | | | | | | | | | | | | |
| **HATEOAS: Hypermedia as the engine of application state 🡪 Jede Resource verfügt über eine Link in die neuen Zustände** | | | | | | | | | | | | | | | |
| **Responsive Layout: Media Queries (PX sind CSS Pixel) Visual Viewport = sichtbar, Layout Viewport=komplett** | | | | | | | | | | | | | | | |
| Device Pixel Ratio ist das Verhältnis zwischen physischer und logischer Pixelgrösse. (DP Ratio = CSS Px / Device Px) | | | | | | | | | | | | | | | |
| @media [screen|print]([width|height|device-x|min-x|max-x]: 10em), (orientation: landscape) {} | | | | | | | | | | | | | | | |
| @media (min-width: 700px), handheld and (orientation: landscape) { } | | | | | | | **or** (default), **and**, **not**, **only** | | | | | | | | |
| <**link rel="stylesheet" media="[print|screen] and (min-width:30em)" href="…"** /> <**meta name="viewport" content="width=device-width, initial-scale=1"**> //~~user-scalable=0~~ | | | | | | | | | | | | | | | |
| **Flexbox: Parent Properties** | | | | | | | | | | | | **Flexbox: Child Properties** | | | |
| **display**: flex;  **flex-direction**: row | row-reverse | column | column-reverse;  **flex-wrap**: nowrap | wrap | wrap-reverse;  **flex-flow**: ‘flex-direction’ || ‘flex-wrap’  **justify-content**: flex-start | flex-end | center | space-between | space-around; //**x-axis**  **align-items**: flex-start | flex-end | center | baseline | stretch; // **y-axis**  **align-content**: flex-start | flex-end | center | space-between | space-around | stretch; | | | | | | | | | | | | **flex-grow**: 0 // default  **order**: 1;  **flex-shrink**: 1 // default  **flex-basis**: 20% | auto;  **flex**: none | ['flex-grow'|'flex-shrink'? || <'flex-basis'> ]  **align-self**: override align-items | | | |
| **Responsive Layout Patterns (Desktop:** Pixel-Grösse wird verändert, **Mobile:** Kompletter Viewport wird vergrössert) | | | | | | | | | | | | | | | |
| Mostly Fluid | | | Column Drop | | Layout Shifter | | | | | | | | | Reflow | | |
| Expand | | | Sidebar/SidePic | | Master dann Detail | | | | | | | | | Off Canvas | | |
| Off-Screen Menu | | | Probleme/**Anti-Pattern**:  zu lange/kurze Zeilen (a,b)  nicht alles sichtbar (b)  schlechte Bildqualität (b,c)  zu untersch. Ansichten für versch. Devices | | | | | | | | | | | | | |
| **Tipps für Fehlermeldungen** | | | Nahe beim Fokus des Nutzer, Keine Beschuldigungen, Keine Fehlercodes, Hilfestellungen | | | | | | | | | | | | | |
| **SCSS / «SASS» : CSS Preprocessor (Mixins vor Extend, Mixins falls Parameter, Extends nur für thematische Vererbung (weniger Code)** | | | | | | | | | | | | | | | |
| Variables | | | $color: #FFF; body, a:#{*$state*} { background-color: $color; } | | | | | | | | | | **Calculations:** + - \* / % | | |
| Nesting / Parent Selektor & | | | ul { li { color: black; } **&**.myClass li { color: red; } } (SCSS) 🡪 ul li {} und ul.myClass li {} (CSS) | | | | | | | | | | | | |
| Partials / Import | | | Datei: \_name.scss | | | | | | @import ‘name’; | | | | | | |
| Mixins | | | @mixin txtCol($col) { color: $col; } | | | | | | li { @include txtCol(red); } | | | | | | |
| Exends / Inherit | | | .err { color: red; } .pro{ @extend .err; more:rules; } 🡪 .err, .pro {color:red;} .pro{more:rules;} | | | | | | | | | | | | |
| Placeholder | | | **%**icon{..} .error-icon{@extend %icon; /\*specific \*/;} .info-icon{@extend %icon; /\*specific \*/;}  🡪.error-icon, info-icon{ /\*both\*/ } .error-icon{/\*specific \*/;} .info-icon{/\*specific \*/;} (CSS) | | | | | | | | | | | | |
| Funktionen | | | @function name($param) {return $param + 1px;} | | | | | | | **Units are important!** 10px\*10px → err | | | | | |
| **Module (High Cohesion, Low Coupling) → Kein global namespace Pollution** | | | | | | | | | | | | | | | |
| Common-JS (nur Node.js) | | | let myApi = *require*(**'./myapi'**); module.exports = { myVar, myFunc } | | | | | | | | | | | | |
| IIFE  (Revealing Module Pattern) | | | (**function** () { **class** Counter { **window**.**game** = **window**.**game** || {};  **window**.**game**.**Counter** = Counter; })(); | | | | | | | | | | | | |
| RequireJS | | | require([**'./myModule'**], **function**(Counter) {  **const** counter = **new** Counter();} define([],**function**(){**class** Counter{..} **return** Counter; }); | | | | | | | | | | | | |
| ES6 | | | **export default** MyModule; *// bei TypeScript* **class** *vor MyModule* **import** MyModule **from './myModule'**; | | | | | | | | | | | | |
| **Typescript (\*.ts) (Pre-Compiler, kein Runtime Checking → Kompiliert JS Code (ES3)) tsconfig.json → target=es6** | | | | | | | | | | | | | | | |
| Es gibt nur function overriding und kein overloading! | | | | | | | | | | | | | | | |
| Wird an Array mit Variablen des selben Types erstellt, wird implizit nur dieser Typ erlaubt **let** myVar :any | :number | :string | :boolean | :myClass;  **let** myArray : number[] = [1,2,3]; **let** myTupel:[string, number]; **myTupel**=[**"a"**, 42]; **let** genericArray : Array<number> = [1,2,3,4] rsp. new Array<MyClass>(); **function** *sum*(n1:number, n2:number):number|:void {...} **// param, return-type** **function** *combine*(sn:number|string=**''**,**optional?**:number|string):string {} | | | | | | | | | | | | | | | |
| **private myvar**: number; *// editor = private, interpretiert = public* **public static** readonly **MY\_CONST** = **'mystring'**; *// konstante* | | | | | | | | | | | | | | | |
| **enum** Color {Red = 1, **Blue**=30, **White**=20}; **let** c : Color = Color.White; | | | | | | | | | | | | | | | |
| **interface** Point { readonly x: number; readonly y: number; } | | | | | | | | | | | | | | | |
| abstract **class** Animal {  **private \_name** : string;  **private static** *readonly* **MY\_CONST** : number = 42;  constructor(name:string) { **this**.**\_name** = name; }  **get** name() : string { **return this**.**\_name**; }  **set** name(val : string) { **this**.**\_name** = name; }  move(distance:number = 0) : **void** { .. } } | | | | | | | **class** Snake **extends** Animal **implements** IMove{  **readonly numberOfLegs**:number = 0;  constructor (name:string){  **super**(name); }  move(distance:number = 0) :**void** {  **super**.move(distance); }} | | | | | | | | |
| Wird kein **this** verwendet, wird die Variable **zuerst** global gesucht. Ansonsten wird nicht global gesucht. | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Testing: Mit Jasmine basierend auf User Stories (🡪 BDD: Behaviour Driven Development)** | | | | | | | | | | | | |
| **User Stories** | | As an [role], I want to [action] so that [role] can give me money | | | | | | | | | | |
| **Smells** | | *- Hard-to-Test Code* (poorly written); - *Production Bugs* (too many bugs);  *- Fragile Test* (Test depends on to much other code);  *- Erratic Test* (Random fails); - *Developers Not Writing Tests* | | | | | | | | | | |
| - Zu viel Asserts pro Test - Unübersichtlicher Test - Langsame Tests(HTTP-Call) -Unterscheidung Prod/Test Env  - Duplicated Code (before each) - Keine if/else in Tests erlaubt | | | | | | | | | | |
| **let *Transaction*** = require(**"**./**transaction");** *describe*(**"A new transaction"**, **function**() {  beforeEach(**function**() { *// Methodstub for Date.now()  spyOn*(Date, **'now'**).andReturn(**new** Date(**"2016-11-22T09:49:51.010Z"**));  **let** BankAccountSpy = **class** { withdraw(){} deposit(){} }; *// Bank account spy*  **this**.**accountA** = **new** BankAccountSpy(); **this**.**accountB** = **new** BankAccountSpy();  **this**.**transaction** = **new *Transaction***(**this**.**accountA**, **this**.**accountB**, 25); });  it(**"has transaction date 2016-11-22T09:49:51.010Z"**, **function**(){  *expect*(**this**.transaction.**date**).toEqual(**new** Date(**"2016-11-22T09:49:51.010Z"**)); });  it(**"has an amount of 25"**, **function**() { *expect*(**this**.transaction.**amount**).toBe(25); });  it(**"is not completed"**, **function**(){*expect*(**this**.transaction.**completed**).toBe(**false**); });  *describe*(**"executed"**, **function**() { beforeEach(**function**(){  s*pyOn*(**this**.**accountA**,**'withdraw'**); *spyOn*(**this**.**accountB**, **'deposit'**);  **this**.transaction.execute(); });  it(**"withdraws 25 from account A"**, **function**() {  *expect*(**this**.**accountA**.withdraw).toHaveBeenCalledWith(25); });  it(**"deposits 25 to account B"**, **function**() {  *expect*(**this**.**accountB**.deposit).toHaveBeenCalledWith(25); });  it(**"is completed"**, **function**() {  *expect*(**this**.transaction.**completed**).toBe(**true**); }); }); }); | | | | | | | | | | | | |
| **Stub** | spyOn(Date, 'now').andReturn(currentDate); | | | | **Spy** | | Hat keinen Rückgabewert. Siehe oben. | | | | | |
| **Fake** | Stellvertreterobjekt für richtigen Service | | | | | | | | | | | |
| **Mock** | Stellvertreterobjekt für richtigen Service. Beinhaltet zusätzlich Expects. Verify Funktion prüft Ausführung der Expects | | | | | | | | | | | |
| **UCD-Vertiefung (Struktur: Wo bin ich? / Raster: Gute Hierachie, Affordance / Oberfläche: Wahrnehmung)** | | | | | | | | | | | | |
| **ProgressiveEnhancement** | | | | | Zusätzliche Inhalte je nach Capabilites nachladen | | | | | **Graceful Degradation** | | Alles nutzen |
| **Fehlermeldungen:** Keine Beschuldigungen, Im Fokus anzeigen, Sprechende Namen, Keine Codes, Hilfestellung zur Fehlerbehebung | | | | | | | | | | | | |
| **Interaktionsschritte nach Norman:**  **Kluft der Ausführung**: (Benutzer→Website): Intention → Handlungsplanung → Ausführung (**Visibility, Affordance**)  **Kluft der Evaluation:** (Website→Benutzer): Wahrnehmung → Interpretation → Bewertung (**Feedback**) | | | | | | | | | | | | |
| **Shared Element Transition:** Vom einen zum nächsten Schritt wechseln nur Teile des UI | | | | | | | | | | | | |
| **Wireframing:** abstrakte Art von Prototyping, unterschieden in zwei Dimensionen: **High/Low Fidelity** (visuelle, interaktive Exaktheit) und Partiell vs. Umfassend(Umfang) | | | | | | | | | | | | |
| **Usability Tests: Card Sorting** (Struktur optimieren)  1) Content Repo erstellen (Zielpunkte d. Navigation) 2) Open Card Sort: 5+ Pers aus Zielgrp rekrutieren und Content Elemente in disjunkte Gruppierungen einteilen+Grp benennen lassen 3) Gute Gruppennamen identifizieren 4) Gruppennamen mit Closed Sort validieren: 5+neue Pers., Gruppennamen vorgeben+Content Elemente den Grp zuteilen lassen  **Usability Tests: Tree Testing:** 1) Aktuelle Baumstruktur(Site-Map) aufnehmen 2) Szenarien zur Erreichung von Zielen stellen + 3)Testen lassen | | | | | | | | | | | | |
| **Objektorientierung: Prototypen, Vererbung, private Methoden, private Member** | | | | | | | | | | | | |
| **let** Parent = (**function**() {  **function** *Parent*(cInit) {  **this**.**count** = cInit;  **this**.myFunc = **function**() {  **return this**.**count**();  }  }  *Parent*.**prototype**.inc = **function**(){}  **return** *Parent*; })**()**; // let p = new Parent(10); | | | | | **let** Derived = (**function**() { **function** *Derived*() { *// sub*  Parent.call(**this**, 0); *// super* }  *// v1: constructor intheritance  Derived*.**prototype** = **new** Parent();  *Derived*.**prototype**.constructor = *Derived*;  **return** *Derived*; })(); *// v2: prototype chain instead of c-inherit* Derived.**prototype**=Object.create(Parent.**prototype**); | | | | | | | |
| **function** *House*(color) {  **let** height = 0; *// private var* **this**.**facadeColor** = color; *// public var* **this**.paint = **function**(newColor) { *// public method*  *repaint*(newColor) }  **function** *repaint*(newColor){ *// private method* **this**.**facadeColor** = newColor; }  Object.defineProperty(**this**, **'height'**, {  get: **function**(){**return** height;},  set: **function** (value) {height = Number(value); }  }); } | | | | | | | | *// No new-keyword* **function** *MyClass*(p){  **if**(!(**this instanceof** *MyClass*)){  **return new** MyClass(p);  }  **this**.**val** = p;  *//...* } **let** test = MyClass(5); | | | | |
| **AJAX** | | | | | | | | | | | | |
| $.ajax({**type**:POST, **dataType**: **"json"**, **url**:**""**,  **data**: **JSON**.stringify(myData), **contentType**: **"application/json"**})  .done(**function**(data, textstatus, jqXHR){..})  .fail(**function**(data, textstatus, errorThrown) {..}) .always(**function**(data) {..}) | | | | | | | | | | | | |
| $.get(**"url"**, **function**(data){..}) | | | | | | $.post(**"url"**, myData, **function**(data){}) | | | | | | |
| **1: JS Code Injection** | | | **3. XSS: Cross Site Scripting** | | | **8. Cross Site Request Forgery** | | | | **2. Broken Auth & Session Mgmt** | | |
| Angreifer kann Server zum Ausführen von ein-geschleustem Code bringen  **Massnahmen:** Kein eval(), setTimeOut(), setInterval(), sodern JSON.parse() und parseInt(). *X-powered-by* Header ausschalten, Node nicht als Root starten. **Angriff:** JS übergeben, das von eval() ausgeführt wird. Z.B while(true) → DoS | | | Angreifer kann Server dazu bringen, Schadcode an Opfer auszuliefern und auf Nutzerseite auszuführen. (Oft über URL-GET-Parameter)  **Massnahmen:** Input Validierung, Escaping, kontext sensitives DataEncoding bei der Darstellung von Nutzer Daten. CSP: Content Security Policy im Header setzen. (Nur Scripts von expliziten Domains erlauben) HTTPOnly-Flag setzen (cookie:{httpOnly:true})  **Angriff:**  name = <script>alert()</script>  Template: Hallo {{{name}}} | | | Ein Web-Site besitzt eine CSRF Verwundbarkeit, wenn bei dieser Site unter Nutzung einer noch nicht ausgelaufenen Session Operationen ausgeführt können. (E-Banking in Tab offen)  **Massnahmen:** Serverseitige Security Tokens. Prüfung des Tokens beim Server. Benutzung der CSRF Middleware: res.locals.csrftoken = req.csrfToke();  **Angriff:** Locken von Benutzer auf eine Seite des Angreifer. Diese Seite enthält ein POST Formular welches submitted wird und Änderungen auf der anderen Seite vornimmt (z.B. neuen admin-Benutzer) | | | | Benutzerspezifische Daten sollten nur für authentisierte Nutzer zugreifbar sein. Angreifer kann SessionID eines nicht ausgeloggten Users verwenden und hat Zugriff auf seine Daten **Massnahmen:** Session Timeout sinnvoll setzen. Verschlüsseln der Verbindung (TLS), Passwörter in Datenbank verschlüsseln. **Angriff:** Sniffen von unverschlüsselten Passwörterr zugreifbten nur für authentisie | | |
| **4. Insecure Direct Obj. Ref.** | | | Über manipulierte Datenschlüssel, kann auf unberichtigte Daten zugegriffen werden. Bsp Link zu UserDetail wird manipuliert. **Massnahmen:** Bei jedem Zugriff überprüfen, ob der aufrufende Berechtigung für die URL hat.  **Angriff:** Parameter in GET URL ändern | | | | | | | | | |