JS OO: Context = "Object's own Memory", access via **this**.
Abnormal Context behaviour: Use JS-Class w/o **new** → Context doesn't change → this (inside of object's methods) = global context.
Auto-Unboxing by calling .valueOf()
Compare/Call/Copy by **reference**.

```
function House (color) {                    // class definition, constructor function
  var self = this;                          // store this pointer on the scope
  self.facadeColor = color;
  self.paint = function(newColor) {         // method definition
    self.facadeColor = newColor;            // do more paint stuff here, colorize windows, etc..
  };
}

var whiteHouse = new House("white");
whiteHouse.paint("beige");                  // call method directly on object; works in any case
var paintWhiteHouse = whiteHouse.paint;     // copy pointer of function paint
paintWhiteHouse("red");                     // call function without object; works now
```

```
function House (color) {                    // class definition, constructor function
  var self = this;                          // store context on the scope
  var height = 0;                           // private field definition
  self.facadeColor = color;                 // public property definition
  self.paint = function(newColor) {         // public method definition
    repaint(newColor);
  };
  function repaint() {                      // private method definition
    self.facadeColor = newColor;
  }
  Object.defineProperty(self, 'height', {   // property definition with accessors
    get: function() { return height; },
    set: function(value) { height = Number(value); }
  } );
}
```

It's a trap! JS Object.prototype

```
o = {};
o.p1 = 7;
Object.prototype.p2 = 4;
o.p2;       // 4
Object.prototype.myFun = function()
{ return this.p1 * this.p2; };
o.myFun();  // 28
o2 = {};
o2.myFun(); // NaN (wegen p1)
myFun();    // NaN(wegen p1)
p1=99;
myFun();    // 396
```

```
o2.myFun();  // NaN, p1 fehlt
myFun.call(o); // 28
Object.prototype.myFun2 =
function () {return p1 * p2};
myFun2();    // 396
o.myFun2();  // 396
o2.myFun2(); // 396
o2.myFun();  // NaN, p1 fehlt
o2.p1 = -1;
o2.myFun();  // -4
Object.prototype.p2 = -44;
o2.myFun();  // 44
```

Object.prototype.toString: returns String repr. of Obj. Default=[object Object]
Idiom: Immediately-invoked Function Expression (**IIFE**):
;(function() { 'use strict'; /* do stuff here in closure scope */ } ());
Intent: Keep global scope clean, Strict mode applied in controlled environm.

Philipp Christen, Ueli Bosshard, Stefan Schindler, Tobias Stauber

JS OOP Inheritance (Prototype based)

```
function Base() {}        // define the Base Class
Base.prototype.beatBox = function(){ alert ("Bum bum bum"); };

function Sub() {      // define the Sub class
  Base.call(this);          // Call the parent constructor
}
Sub.prototype = new Base();   // inherit Base

// correct the constructor pointer because it points to Base
Sub.prototype.constructor = Sub;

Sub.prototype.beatBox = function() { // replace beatBox()
  alert('Sub stands for');
}
```

`var b = new Base();` `b.beatBox(); // Bum bum bum`	`var s = new Sub();` `s.beatBox(); // Sub stands for`
`b instanceof Base  // true` `b instanceof Sub   // false`	`s instanceof Base   // true` `s instanceof Sub   // true`

JS Constructor-Based Inheritance

```
function Base( myName ) {
  this.name = myName;
  // new Copy of say() for each Object Instance
  this.say = function(){console.log("Hi, I'm " + this.name);};
}
function Sub( myName ) {
  Base.call( this, myName );
  this.sayStuff = function(stuff) { console.log( stuff ); };
}
```

`var b = new Base("Hans");`
`b.say(); // Hi, I'm Hans`
`b.sayStuff("asd"); // Error`
	`var s = new Sub("Paul");`
	`s.say(); // Hi, I'm Paul`
	`s.sayStuff("asd"); // asd`

`b instanceof Base  // true` `b instanceof Sub   // false`	`s instanceof Base  // false` `s instanceof Sub   // true`

CSS Einheiten:

px | %	Pixel* bzw. Prozent
pt | pc	Print Punkte bzw. Pica
em	hängt vom <partent> ab
rem	hängt von `html { font-size: 17px; }` ab
cm | mm | in	Centi-, Milli-Meter oder Inch
vw | wh	Relative to 1% of the width | height of the viewport. Viewport = the browser window size.
vmin | vmax	Relative to 1% of viewport's* smaller dimension

* Pixels are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.
Umrechnung: 1in = 2.54cm = 25.4mm = 72pt = 6pc

calc([wert] [operator] [wert])
Operatoren: * / + -

Flex:

Container:

display: flex; // init flex layout
flex-direction: row | row-reverse | column | column-reverse;
flex-wrap: nowrap | wrap | wrap-reverse;
flex-flow: <shorthand for flex-direction and flex-wrap>
justify-content: flex-start | flex-end | center | space-between | space-around; // alignment along ↔ axis
align-items: flex-start | flex-end | center | baseline | stretch; // alignment along. Baseline = text-bottom is aligned
align-content: flex-start | flex-end | center | space-between | space-around | stretch; // alings container's lines if there's extra space in ↕ axis

Item:

order: <integer>;
flex-grow: <number (default: 0)>; // relative share of remaining space (all have 1 → even distribution), 0 = doesn't grow
flex-shrink: <number (default: 1)>;
flex-basis: <length> | auto; // size that the element takes before remaining space is distributed
flex: <shorthand for flex-grow, flex-shrink and flex-basis>
align-self: auto | flex-start | flex-end | center | baseline | stretch; // override **align-items** of container

---

Media Queries:

```
<link rel="stylesheet" type="text/css" media="screen" href="screen.css">
@media screen {}   oder   @media print {}
@media ([width|min-width|max-width]: 375px) {}
@media ([device-width|min-device-width|max-device-width]: 50px){}
/* natürlich auch mit height statt width */
@media (min-width: 20em) and (max-width: 30em) {} /* AND */
@media (max-width: 10em), (min-width: 20em) {} /* OR */
```

SASS: (SASS = SCSS ohne ; oder {} aber mit Einrückung)

```
@import 'otherFile'; // imports _otherFile.scss
$color: #FF0000; $number: 200; $unit: 42px;

@mixin addSymbol($position : before, $symbol: "<"){
  &:#{$position}{ content: $symbol; }
} // before and '<' are default values!
.symbol{ @include addSymbol(after, "?"); } //adds ? after elem
.bar {@extend .foo; } // aus .foo{...} wird .bar, .foo {...}
```

Custom Functions

```
$grid-width: 40px; $gutter-width: 10px;
@function grid-width($n) {  /* Definition */
  @return $n * $grid-width + ($n - 1) * $gutter-width;
}
#sidebar { width: grid-width(5); } /* Usage */
```

Typescript (Super-Set von JS → All valid JS is valid TS)

Variables:

```
var isDone: boolean = false;
var height: number = 6;
var name: string = "Little Bobby Tables";
var list:number[] = [1, 2, 3];
var list:Array<number> = [1, 2, 3];
enum A {one, four=4, five}; //default # starts at 0
var notSure: any = 4; // geht auch als (mixed) Array!
```

Functions:

```
// myAdd has the full function type
var myAdd = function(x: number, y: number): number {
return x+y; };
// 'x' and 'y' have the type number
var myAdd: (base:number, inc:number)=>number =
    function(x, y) { return x+y; };

function concat(first: string = "A",second?: string){
  return ( second ? first + " " + second : first );
} // first hat Default-Wert "A", second ist optional

class Employee {
  constructor(init?: string) { this._name = init; }
  private _name: string;
  get name(): string { return this._name; }
  set name(newName: string){ this._name = newName; }
}
var e1 = new Employee(); e1.name = "Anne";
var e2 = new Employee("Bob"); // uses opt. CTOR param
```

Employee wird kompiliert zu:

```
var Employee = (function () {
  function Employee(init) { this._name = init; }
  Object.defineProperty(Employee.prototype, "name", {
    get: function () { return this._name; },
    set: function (newName){ this._name = newName; },
    enumerable: true, configurable: true
  }); return Employee;
})();
```

ASP.NET

OWIN: Open Web Interface (.NET), entkoppelt Web Server/FW

Katana: Impl. of OWIN, bietet Pipelines (Middleware in Express)

Authentifizierung: Wer bin ich? Autorisierung: Was darf ich?

Razor: Template Engine für HTML

Angular V2 (Wenn Controller/Services/... separiert)

myCtrl.js

```
var MyCtrl = function($scope, MySrvc) {...};
```

mySrvc.js

```
var MySrvc = function() {...}; // no dependencies
```

myApp.js

```
var App = angular.module('MyApp',[]);
App.service('MySrvc', MySrvc); // register service
MyCtrl.$inject = ['$scope', 'MySrvc']; //inject stuff
App.controller('MyCtrl', MyCtrl); // register ctrl
angular.bootstrap(...);
```

AngularJS, "Random" Example

modules.js:

```
var randomApp = angular.module('RandomApp', ['ngRoute']);
randomApp.config(['$routeProvider', function($routeProvider) {
  $routeProvider.when('/randoms', {
    templateUrl: "random.html", controller: "RandomController"
  });
  $routeProvider.otherwise({ redirectTo: '/randoms' });
}]);
randomApp.service("RandomService", ["$http", function($http){
  var RandomService = {};
  RandomService.getRandom = function(successCallback) {
    var rBool = Math.round(Math.random()) === 1;
    $http.get("http://echo.jsontest.com/rb/" + (rBool ? "Ja":"Nein"))
   .then(function(response){successCallback(response.data.rb);
    });
  };
  return RandomService;
}]);
randomApp.controller('RandomController', ["$scope", "RandomService",
function($scope, RandomService) {
  $scope.nrOfRands = 2; $scope.randoms = [];
  $scope.again = function() {
    $scope.randoms = [];
    for( var i = 0; i < $scope.nrOfRands; i++ ) {
      RandomService.getRandom(function( response ) {
        $scope.randoms.push( response );
      });
    }
  };
  $scope.again();
}]);
```

random.html:

```
<input data-ng-model="nrOfRands"/>
<div data-ng-repeat="rand in randoms track by $index">{{ rand }}</div>
<button data-ng-click="again()">Nochmals</button>
```

index.html:

```
<!DOCTYPE html><html><head><meta charset="utf-8">
  <title>Random Stuff yo</title>
  <script src="angular.min.js"></script>
  <script src="angular-route.min.js"></script>
  <script src="modules.js"></script>
</head><body>
  <div data-ng-app="RandomApp"> <div data-ng-view></div> </div>
</body></html>
```

Testing (Jasmine)

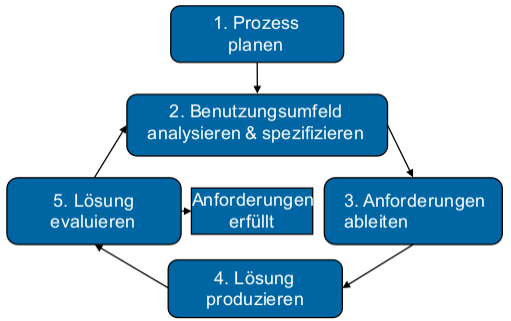
```
describe('RandomController', function() {
   var $httpBackend, $rootScope, createController;
   beforeEach(module('RandomApp')); // Set up the module
   beforeEach(inject(function($injector) {
     $httpBackend = $injector.get('$httpBackend');
     mockReply = $httpBackend.when('GET','/random').respond({random:'Ja'});
     $rootScope = $injector.get('$rootScope');
     var $controller = $injector.get('$controller');
     createCTRL = function() {
       return $controller('RandomController', {'$scope':$rootScope});
     };
   }));
   afterEach(function() {
     $httpBackend.verifyNoOutstandingExpectation();
     $httpBackend.verifyNoOutstandingRequest();
   });
   it('should get two bools from the backend', function() {
     $httpBackend.expectGET('/random');
     var controller = createCTRL();
     $httpBackend.flush();
     expect($rootScope.rands).toBe(['Ja', 'Ja']);
   });
});
```

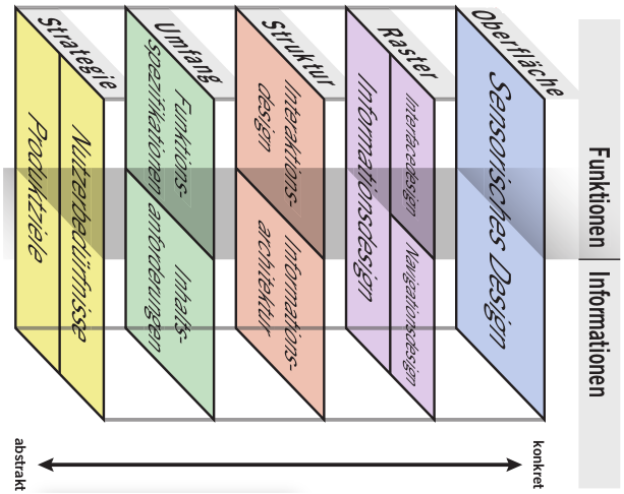
Usability und Co

Graceful Degradation: Nutzt alle Features. Nicht vorhanden? → Es muss dem User eine sinnvolle Alternative geboten/darauf hingewiesen werden.

Progressive Enhancement: Soll für alle zugänglich sein. Start mit Basics (no JS/Media Queries), Extras mit CSS und JS nachladen, wenn supported.

User Centered Design Process





Cognitive Walkthrough Example: Szenario = Ziel = Aufgabe (meist > 1)

Vorgeschichte: Konferenz geplant im ABC Center (1.4.16) → Suche Ort für Galadiner 50-200 Personen. *Auslöser*: Broschüre fertigstellen.

Ziel: Entfernung zum ABC Center. Bilder, Grösse, Verfügbark., Preis. *Startpunkt im UI*: Homepage. *Benutzerprofil*: Desktop-user, 40J, erfahren

Kriterien gem. Stone et al:

Visibility: Nächster Schritt zum Ziel ist sichtbar.

Affordance : "Begreifbarkeit" Bedienung / Funktion von Control ist einsichtig (erkennbar/erinnerbar). Aktionsresultat ist vorhersagbar. (auch "Signifiers")

Feedback: Es ist klar, was passiert (ist), (z.B. durch Animation

Simplicity: Nicht mehr als nötig für die Aufgabe (verbunden mit Visibility)

Structure: Logische und konsistente Organisation (verbunden mit Visibility)

Consistency: Vorhersagbarkeit durch Konsistenz (verb. mit Visib.&Aff.)

Tolerance: Fehler vermeiden, Wiederherstellung vereinfachen (Feed.&Aff.)

Accessibility: Design für alle Personengruppen & Situationen

Usability Test: keine Schritte, sondern nur für Nutzer relevante Ziele (nicht: einloggen, Einstellungen vornehmen etc.), Echte Benutzerziele (nicht "Sichern sie Ihr File"), Keine "Keywords" verraten, z.B. mit Bildern arbeiten. Bsp: Sie befinden sich gerade mit Ihrer Freundin, die Hunger auf Pizza hat, in der Stadt St. Gallen. Verwenden Sie die App XYZ, um die beste Pizzeria der Stadt zu finden.