BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

# Backend - Vertiefung

## Multimedia Engineering II
## WS 13/14

Sven Spielvogel

# Backend - Vertiefung

- Was ist eine API?
- Welche Merkmale besitzt eine API und welchen Einfluss haben diese?
- Welche negativen Aspekte gibt es in Bezug auf eine API zu beachten?
- Was bedeutet RESTful?
- Welche „Regeln“ gibt es für eine RESTful-Schnittstelle?
- Ist JSON immer kleiner als XML?
- Was ist JSON?
- Welche Vorteile hat JSON gegenüber XML?

# Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Grundbegriffe
  - API
  - REST
  - JSON
- Programmierung – PHP
  - Kurzer Überblick
- Datenbankvergleich
    - MongoDB
    - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

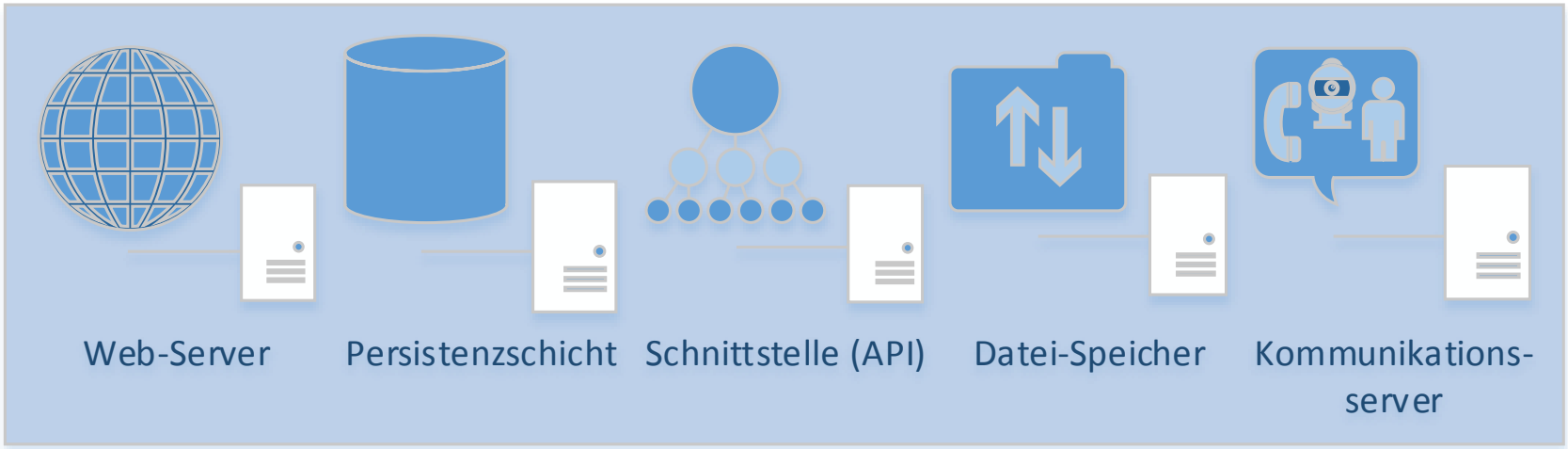
# Letzte Vorlesung

# Backend - Einführung

- Backend
  - Backendtechnologien im kurzen Überblick
  - Installation der Basics
    - Apache, MySQL, PHP
    - Nginx
    - node.js
    - mongoDB
    - Verbindung PHP und mongoDB
    - Verbindung node.js und mongoDB
  - Kleiner Einblick in die Programmierung

# Backend

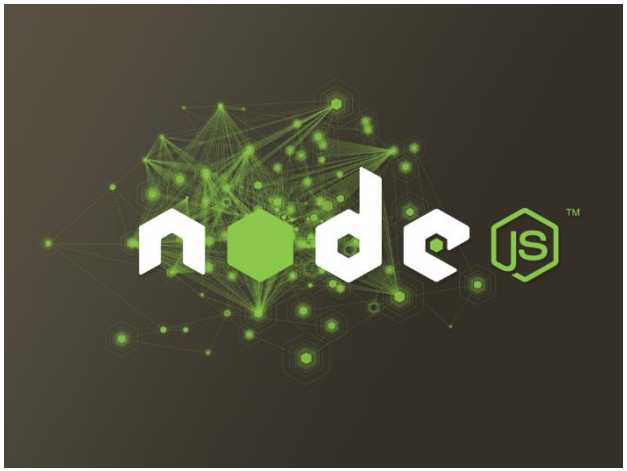
- Aufgaben in unserem Umfeld

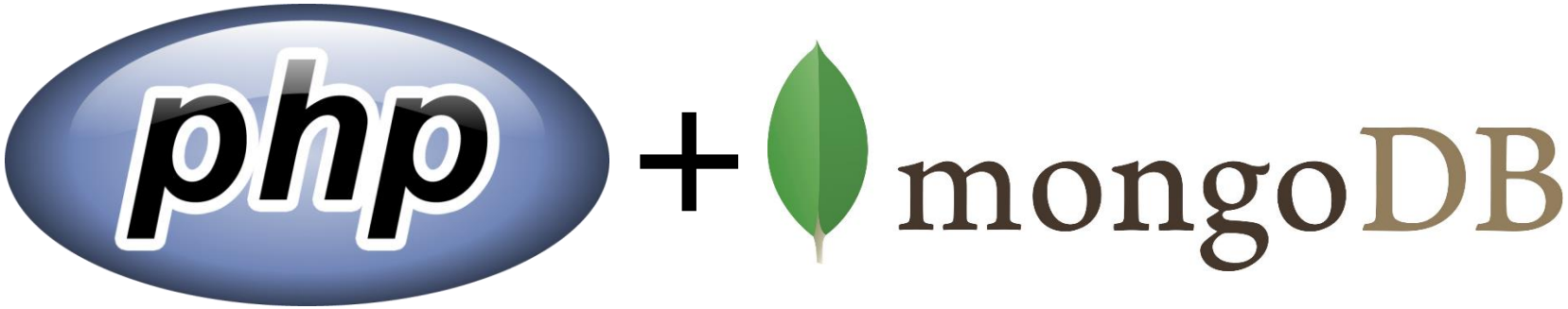


# Installation verschiedener Technologien

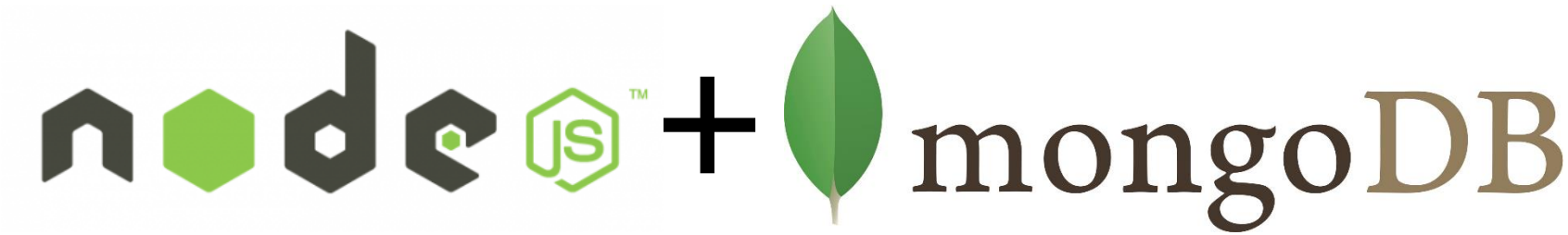





php
+
mongoDB

```php
<?php
try {
	// open connection to MongoDB server
	$conn = new Mongo('localhost');
	// access database
	$db = $conn->test;
	// access collection
	$collection = $db->test;
	// execute query
	// retrieve all documents
	$cursor = $collection->find();
	//print result
	echo json_encode(iterator_to_array($cursor));
	// disconnect from server
	$conn->close();
} catch (MongoConnectionException $e) {
		die('Error connecting to MongoDB server');
} catch (MongoException $e) {
		die('Error: ' . $e->getMessage());
}
?>
```


node
JS
+
mongoDB

- MongoJS

```
var express = require('express');
var mongojs = require('mongojs');
var app = express();
var db = mongojs('test', ['test']);
app.get('/', function(req, res){
        db.test.find(function(err, docs) {
                console.log(docs);
                res.send(docs);
        });
});
app.listen(8080);
```

# Grundbegriffe

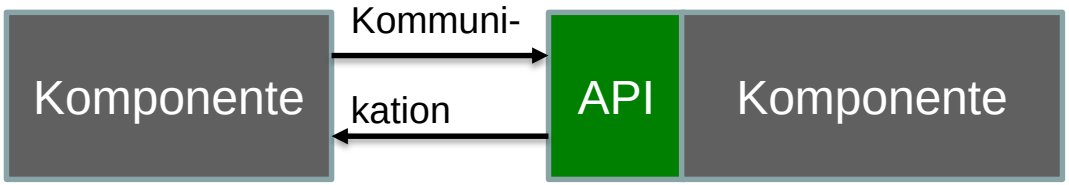
# Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Grundbegriffe
  - API
  - REST
  - JSON
- Programmierung – PHP
  - Kurzer Überblick
  - Datenbankverbindungen
    - MongoDB
    - MySQL
    - ORM-Mapper
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

# API

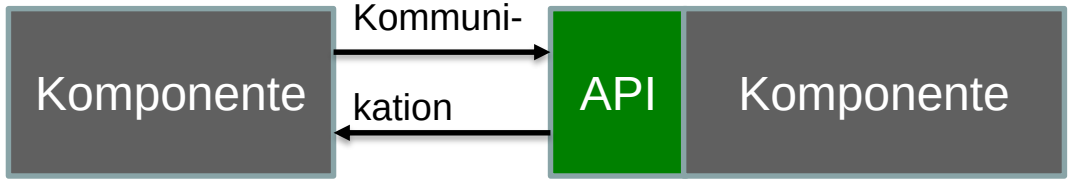
# API

- **A**pplication **P**rogramming **I**nterface
  - Schnittstelle zur Anbindung verschiedener Hard- und Software-Komponenten untereinander
  - Ermöglicht die Kommunikation und dient somit als Bindeglied



# Merkmale einer API

- Unabhängige Schnittstelle
  - →Kann von verschiedenen Systemen genutzt werden
- Einheitliche Kommunikation
  - → I/O-Zuverlässigkeit
- Gute Dokumentation!!
  - → Wichtig für den Verkauf des Produktes



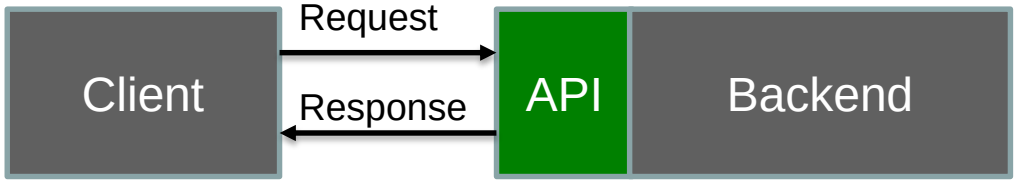
# API – Der Begriff im Bezug auf das Web

- Unser Bereich: Web
  - Schnittstelle zu einer fremden Software

→Datenbankschnittstelle

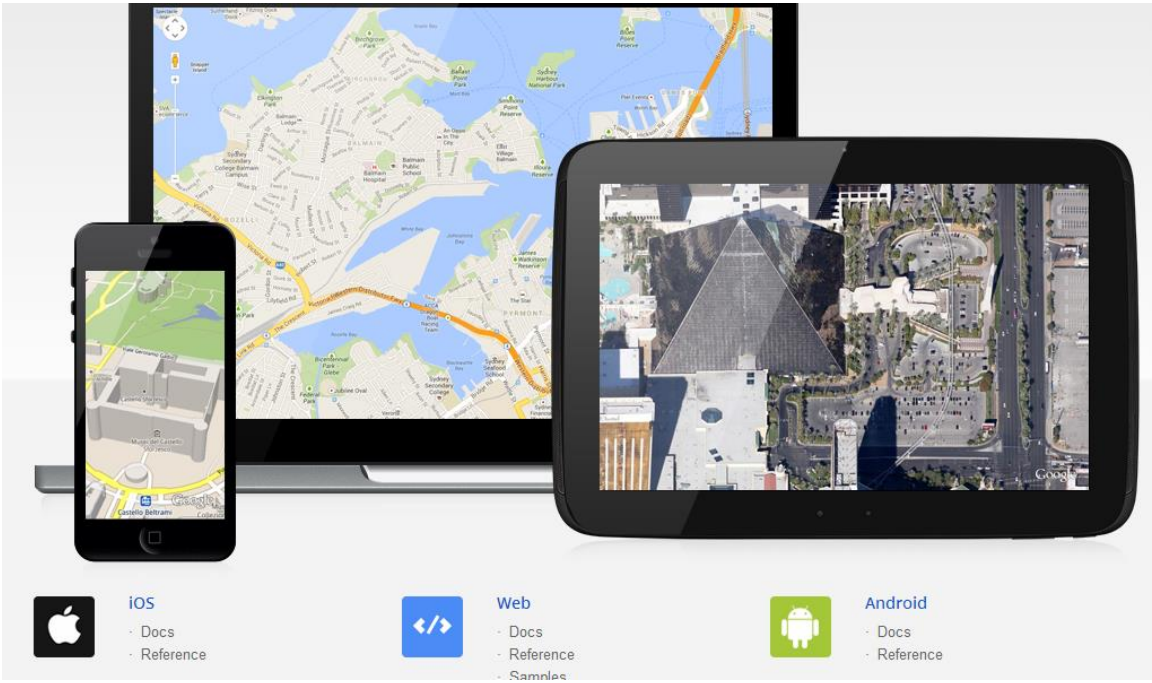
→Google Maps API

→Facebook, Dropbox, …



# Beispiel Google Maps API

- API




- https://developers.google.com/maps/

# Beispiel Google Maps API

- Um Zugriff auf die API zu erhalten, benötigt man einen API Key

```
<script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/
      js?key=API_KEY&sensor=SET_TO_TRUE_OR_FALSE">
</script>
```

- https://developers.google.com/maps/documentation/javascript/tutorial

# Beispiel Google Maps API

- Entwicklung mit Dokumentation, bzw. SDKs
  - Enthalten Code-Snippets oder ganze Beispiel-Applikationen auf Basis von „Hallo Welt“

# API forever?

- APIs sind nicht für die Ewigkeit
  - API wird restriktiv
  - API wird eingestellt
  - API wird kostenpflichtig
- API wird ohne Entsprechender Versionierung umgestellt
  - Alte Implementierung der API

# REST

# Vor REST

```
/api/customers.php?
action=listorders&cid=33245&items=all
```

Gib mir alle Bestellungen des Kunden 33245

# REST

```
/api/customers/33245/orders
```

Gib mir alle Bestellungen des Kunden 33245

# REST

- Representational State Transfer
  - Einheitliche Uniforme Schnittstelle
  - Lesbares Targeting in der URL

Was bedeutet das im Detail?

# REST – Ein paar „Regeln“

- Verwendung von Substantiven
  - Plural > Singular

```
/api/oders/12/items
```

  - Möglichst spezifisch

```
/api/documents
/api/orders
```

# REST – Ein paar „Regeln“

- Eindeutige Adressierung (URL)
  - Auflistung aller Kunden

    `/api/customers`
  - Auflistung aller Bestellungen

    `/api/orders`
  - Auflistung der Bestellung mit der Nummer „1234“

    `/api/orders/1234`

# REST – Ein paar „Regeln“

- Suchparameter nach dem „?“

```
/orders?color=red&state=open
```

# REST – Ein paar „Regeln“

- HTTP-Operationen (GET,PUT,POST,DELETE)

| HTTP | CRUD |
|---|---|
| `GET` | Read |
| `POST` | Create |
| `PUT` | Update |
| `DELETE` | Delete |

# REST – Ein paar „Regeln“

- HTTP-Operationen (GET,PUT,POST,DELETE)

| Ziel | POST<br>create | GET<br>read | PUT<br>update | DELETE<br>delete |
|---|---|---|---|---|
| `/orders` | Neue Bestellung erstellen | Auflisten der Bestellungen | Alle Bestellungen aktualisieren | Alle Bestellungen löschen |
| `/orders/1` | Fehler | Details für Bestellung 1 | Aktualisieren - wenn nicht vorh. Fehler | Bestellung 1 löschen |

# REST – Ein paar „Regeln“

- Abgeschlossen/Zustandslos
  - Keine Verkettung von Requests
  - Jede Nachricht enthält alle Informationen

- Tipp! Cookie = Zustandsinformation
  - Ggf. Token/API-Key
  - Authentifizierung ggf. mit OAuth2

  http://oauth.net/2/

# REST – Ein paar „Regeln“

- HTTP Zustandscodes nutzen
  - Übernehmen/Bieten die meisten Frameworks
  - jQuery AJAX überprüft Status-Codes

```
200 ok
400 bad request
401 unauthorized
403 forbidden
```

- Bei Fehlern Fehlerbeschreibung in der Antwort

# REST – Ein paar „Regeln“

- Versionierung der API

  `/api/`**`v2`**`/oders/12/items`

  → Wichtig für verschiedene Releases!

Mit der Versionierung wird verhindert, dass bestehende Anbindungen an die API bei einer Änderung die Kompabilität verlieren!

# REST – Ein paar „Regeln“

- Pagination

  `/api/oders?limit=25&offset=50`

  → default

Pagination verhindert bei großen Datenmengen unnötige Ladezeiten und Überlastungen!

# REST – Ein paar „Regeln“

- Ausgabeformat

  `/api/oders.json  oder /api/oders.xml`

  → default je nach API

Größeres Spektrum zur Anbindung.

```
{"next":"JSON"}
```

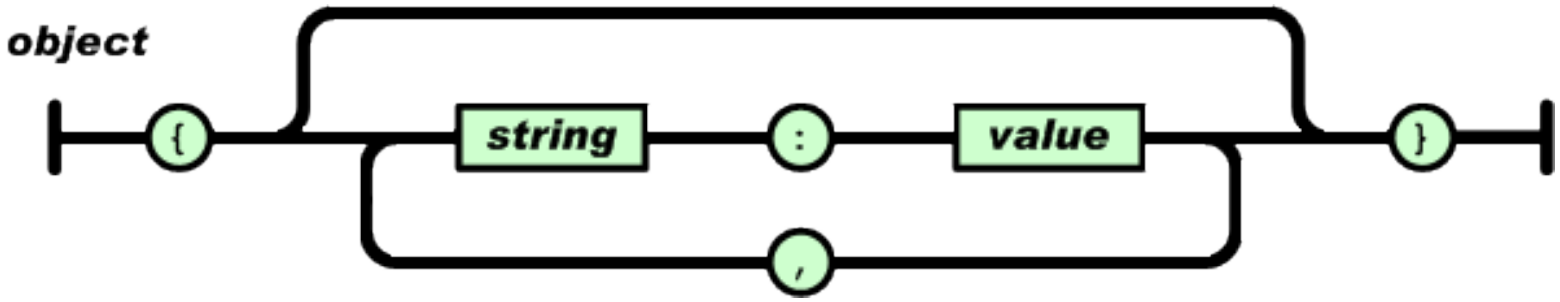
- **J**ava**S**cript **O**bject **N**otation
  - Syntax zum Austausch und Speichern von Text-Informationen
  - Sprachunabhängig
  - Selbstbeschreibend
  - schneller und leichter zu parsen als XML
  - JSON kann größer sein als XML*
  - Parser in nahezu jeder Programmiersprache


* http://wiki.hl7.org/index.php?title=JSON_based_ITS

# **J**ava**S**cript **O**bject **N**otation

```
{
    "employees": [
        { "firstName":"John" , "lastName":"Doe" },
        { "firstName":"Anna" , "lastName":"Smith" },
        { "firstName":"Peter" , "lastName":"Jones" }
    ]
}
```

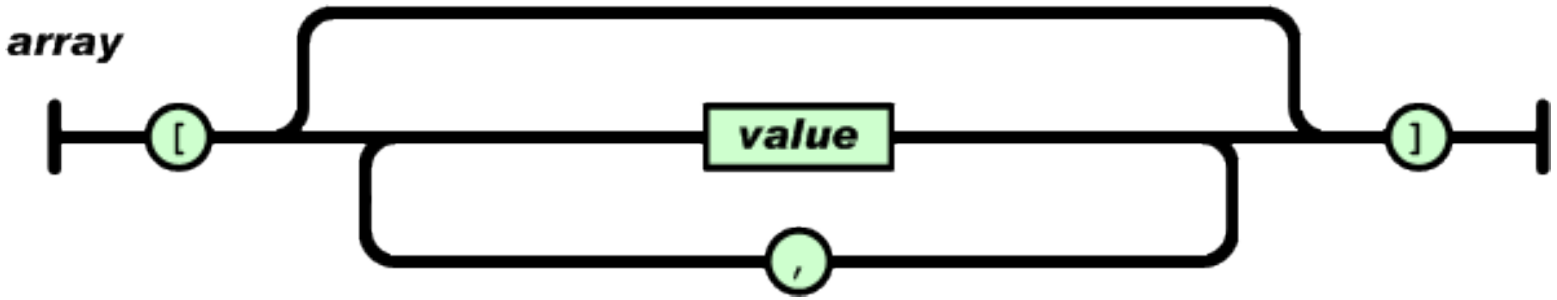
# JavaScript Object Notation




```
{"firstName":"John"}
```

```
{"firstName":"John","lastName":"Doe"}
```

# JavaScript Object Notation

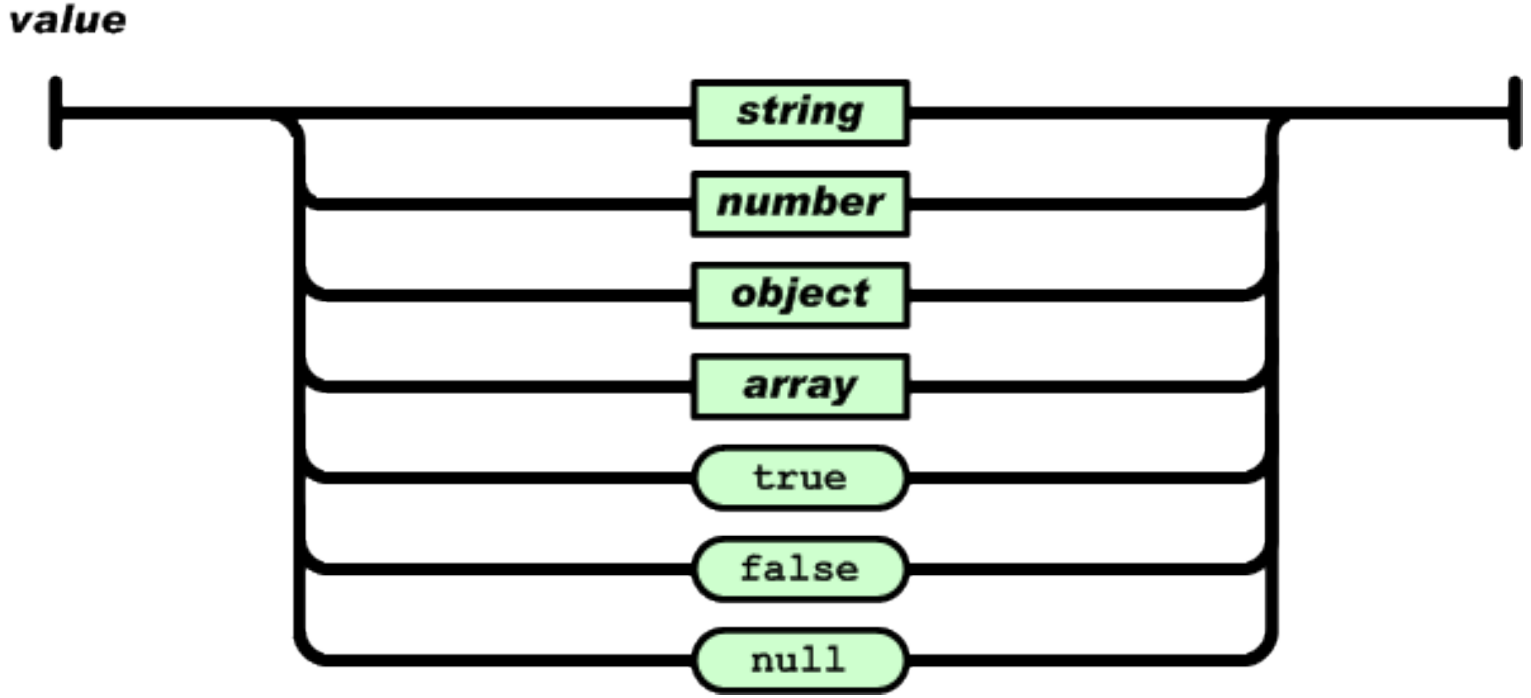



```
{"meinArray":[value]}
```

```
{"meinArray":[value, value]}
```

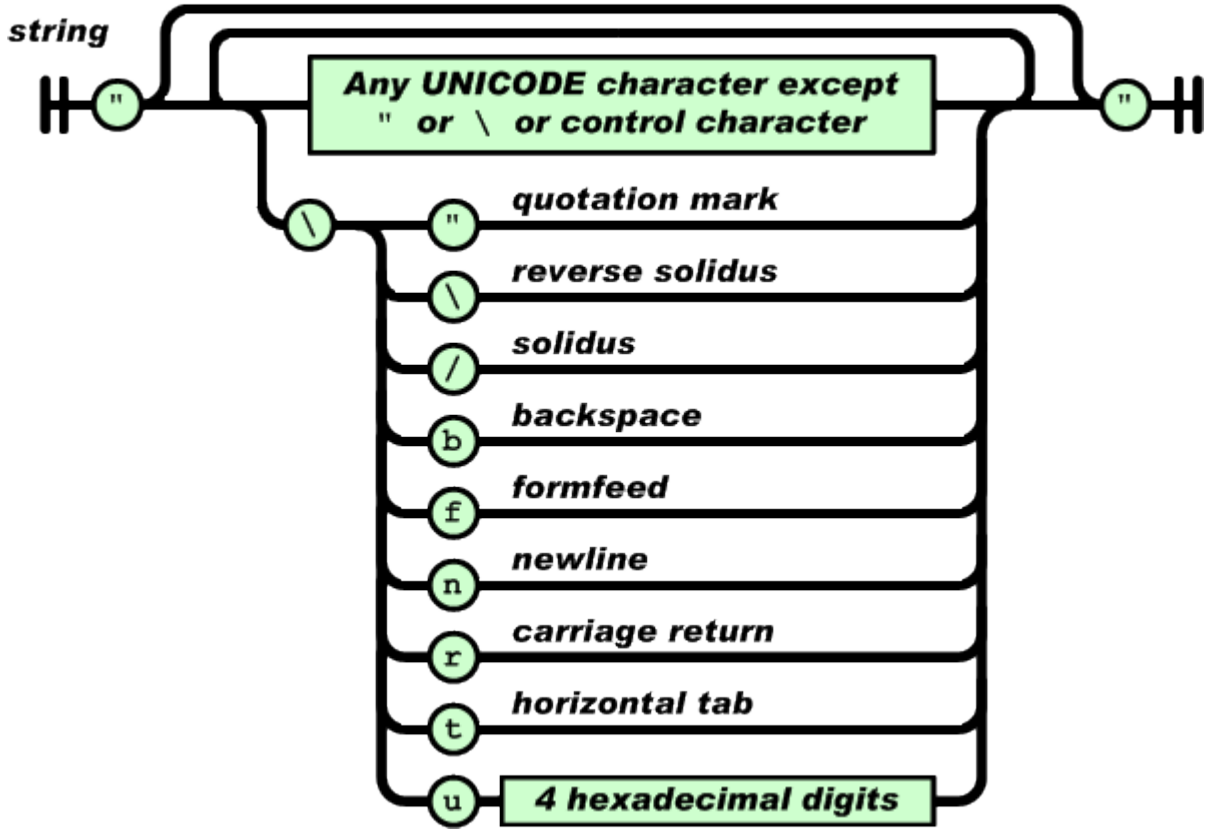
```
{"meinArray":[1, 2]}
```

# JavaScript Object Notation



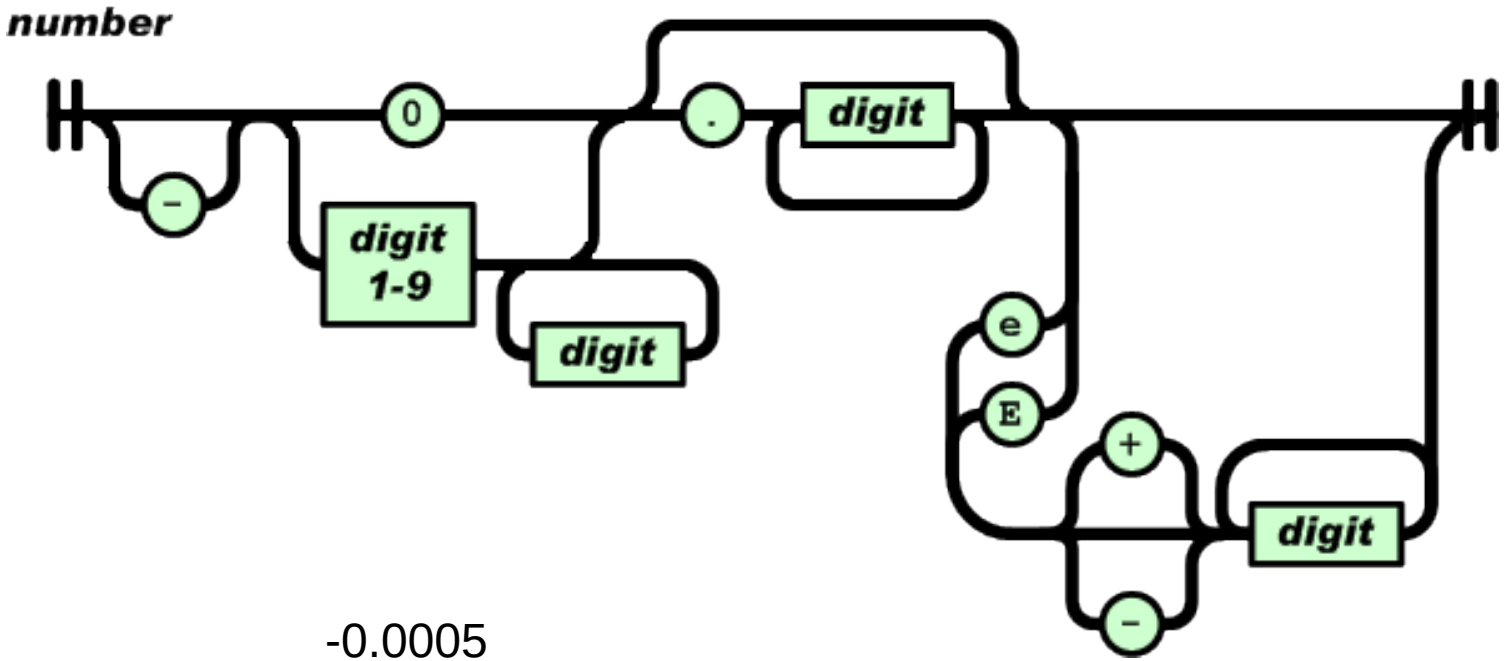

```
{"meinArray":["hi", 2, null, true,{},[1,false]}
```

# JavaScript Object Notation




```
{"meinText":"Hallo \"mein\" Freund, \n Bye"}
```

# JavaScript Object Notation




-0.0005

```
{"meineZahl":-0.5e-3}
```

- Zusammenfassung
  - schnell
    - http://jsperf.com/domparser-vs-json-parse/4
  - gut lesbar
  - schmal
  - ein String
  - kompatibel

# Backend - Vertiefung

- Ein kurzer Rückblick auf die Letzte Vorlesung
- Grundbegriffe
  - API
  - REST
  - JSON
- Programmierung – PHP
  - Kurzer Überblick
- Datenbankvergleich
  - MongoDB
  - MySQL
- RESTful-Schnittstelle
  - PHP – SlimPHP
  - NodeJS – Express
  - Alternativen
- Testen einer API
- Dokumentation

# Programmierung

# PHP

- PHP (rekursives Akronym für PHP: Hypertext Preprocessor)
  - Open Source
- Serverbasierte Skriptsprache

  → wird nur auf dem Server ausgeführt

  → rendert die Ausgabe oder Teile davon

- Opening Tags

```
<?php
  echo "Hello World";
?>
```

- Kurz (benötigt *short_open_tag=true*)

```
<?
  echo "Hello World";
?>
```

- PHP in HTML

```
<!DOCTYPE HTML PUBLIC "-
//W3C/DTD HTML 4.01 Transitional//EN
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Beispiel</title>
    </head>
    <body>

        <?php
           echo "Hallo Welt!";
        ?>

    </body>
</html>
```

# PHP – Primitive Typen

- Vier skalare Typen:
  - Boolean
  - Integer
  - Fließkomma-Zahl (float) (auch double)
  - String / Zeichenkette
- Zwei zusammengesetzte Typen:
  - Array
  - Objekt
- Und zuletzt drei spezielle Typen:
  - Ressource
  - NULL
  - callable

http://php.net/manual/de/language.types.intro.php

- Variablen definieren

```
<?php
  $hello = "Hello World";
  echo $hello;
?>
```

- Arbeiten mit String-Variablen

```
<?
  $hello = "Hello";
  $world = "World";
  echo "Hello $world! Here I am!";
  echo "Hello ". $world ."! Here I am!";
?>
```

- Arbeiten mit Zahlen-Variablen

```
<?
  $one = 1;
  $two = 2;
  echo $one + $two;
  $three  = 4 - $two;
  $three++;
  echo $three;
?>
```

- Konstanten

```
<?php

  define('FOO_BAR','It works!');

  // prints 'It works!'
  $changing_variable = 'bar';
  echo constant('FOO_' . strtoupper($changing_variable));

?>
```

- Methoden

```
<?php
  function foo ($a,$b = 1)
  {
    return 3 + $a - $b;
  }

  echo $foo(1); //3
  echo $foo(5,2); //6
?>
```

# Operatoren

| Assoziativität | Operator |
|---|---|
| keine Richtung | new |
| rechts | [ |
| rechts | ! ~ ++ -- (int) (float) (string) (array) (object) @ |
| links | * / % |
| links | + - . |
| links | << >> |
| keine Richtung | < <= > >= |
| keine Richtung | == != === !== |
| links | & |
| links | ^ |
| links | \| |
| links | && |
| links | \|\| |
| links | ? : |
| rechts | = += -= *= /= .= %= &= \|= ^= <<= >>= |
| rechts | print |
| links | and |
| links | xor |
| links | or |
| links | , |

- Zeilenkommentar

```
<?php
  //Gib Hello World aus
  echo "Hello World";
?>
```

- Blockkommentar

```
<?
  /*
     Dies ist ein Blockkommentar
  */
  echo "Hello World";
?>
```

- Dokumentations-Kommentar

```
<?
  /**
     Diese Funktion foot
     @return
  */
  function foo ()
  {
    return 5;
  }
?>
```

- Dokumentations-Kommentar

```
<?php
class A
{
    // Deklaration einer Eigenschaft
    public $var = 'Hello!';

    // Deklaration einer Methode
    public function foo() {
        echo $this->var;
    }
}

$a = new A();
$a->foo();
?>
```

- Klassen/Objekte und Aufrufe

```php
<?php
class A
{
    // Deklaration einer Eigenschaft
    public $hello = 'Hello!';
    private $world = 'World!';

    // Deklaration einer Methode
    public function foo() {
        return $this->world;
    }
}

$a = new A();
echo $a->hello; //'Hello!'
echo $a->foo(); //'World!'
?>
```

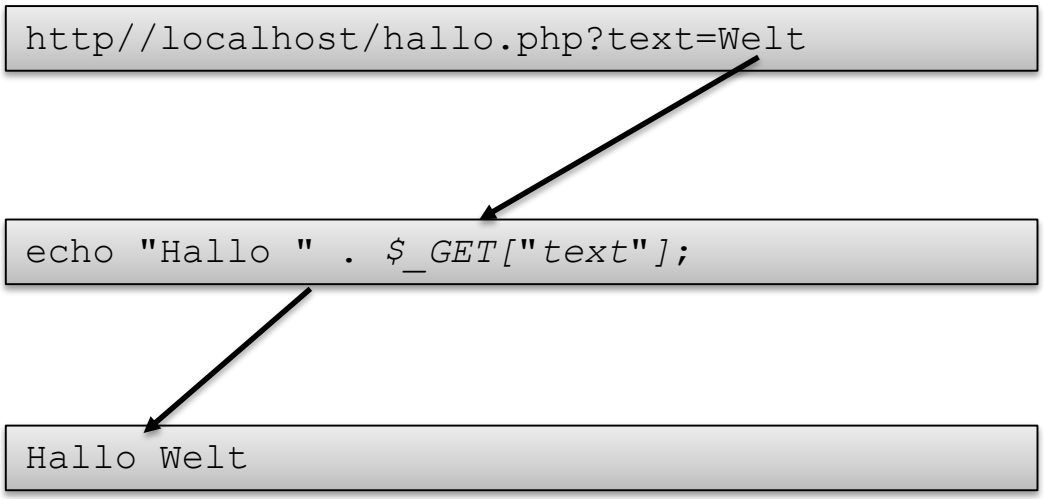
De-/Konstruktor: http://php.net/manual/de/language.oop5.decon.php

- Kontrollstrukturen
  - if/else/elseif

```
if ($a > $b){
  echo "a ist größer als b";
}
```

```
if ($a > $b) {
    echo "a is größer als b";
} elseif ($a == $b) {
    echo "a ist gleich groß wie b";
} else {
    echo "a ist kleiner als b";
}
```

- Sog. Superglobals z.B.
  - *$_GET*
  - → *$_GET["text"]*



- Diese Superglobals sind:
  - *$GLOBALS*
  - *$_SERVER*
  - *$_GET*
  - *$_POST*
  - *$_FILES*
  - *$_COOKIE*
  - *$_SESSION*
  - *$_REQUEST*
  - *$_ENV*

http://www.php.net/manual/de/language.variables.superglobals.php

- Umwandlung Mixed→JSON:
  - json_encode

```
$var = array(true,false, "hallo");
json_encode($var);
```

```
[
    true,
    false,
    "hallo"
]
```

http://php.net/manual/de/book.json.php

- Umwandlung JSON→ Mixed:
  - json_decode

```
$var = "[true,false,`hallo`]";
json_decode($var);
```

```
array(true,false,"hallo");
```

http://php.net/manual/de/book.json.php

# Vielen Dank für die Aufmerksamkeit!