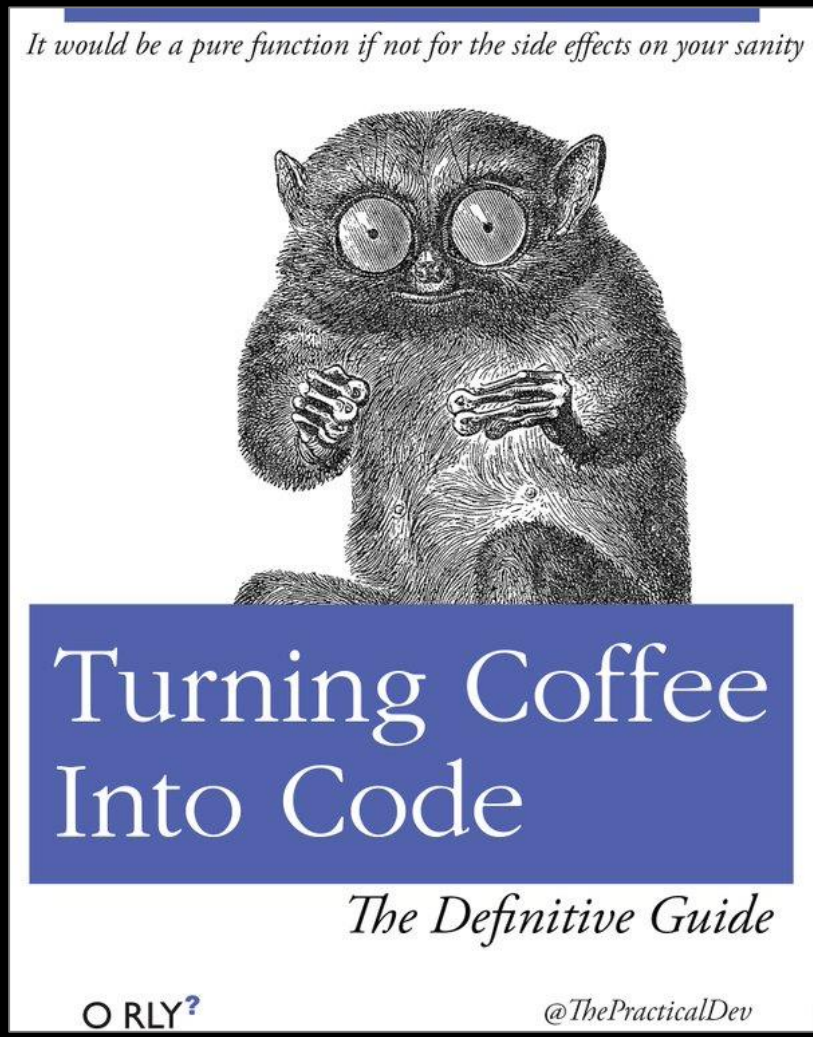
Web Technologies

Web programming (IV)
Web application architecture
application servers
PHP language and
environment



Dr. Sabin Corneliu Buraga – profs.info.uaic.ro/~busaco/

“Poor is the pupil
who does not surpass his master.”

Leonardo da Vinci

By what means
a Web application can be implemented?

implementation

Web application server

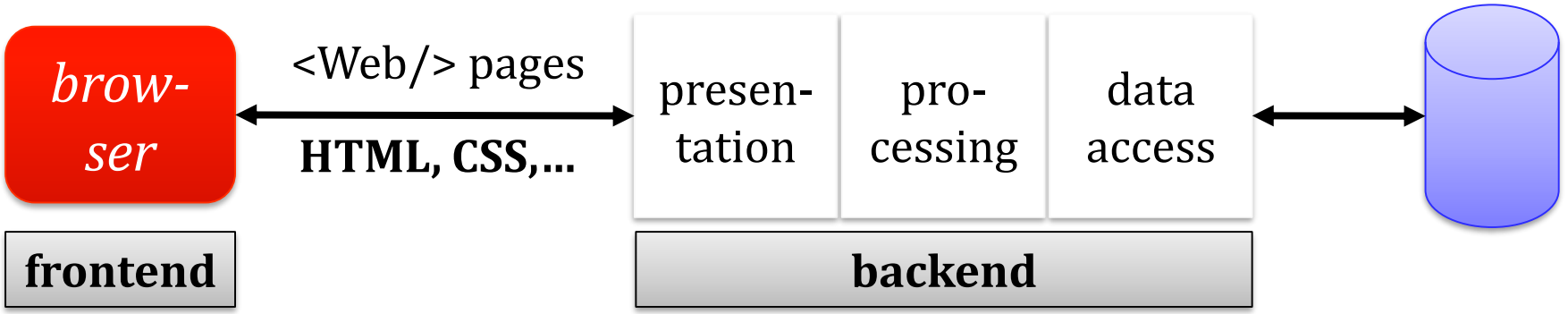
goal:
optimizing the development processes
of complex Web applications

could encourage or impose an architectural approach for Web application development – e.g., MVC or variants

Web application architecture: **classic MV* approach**

dumb client

fat server

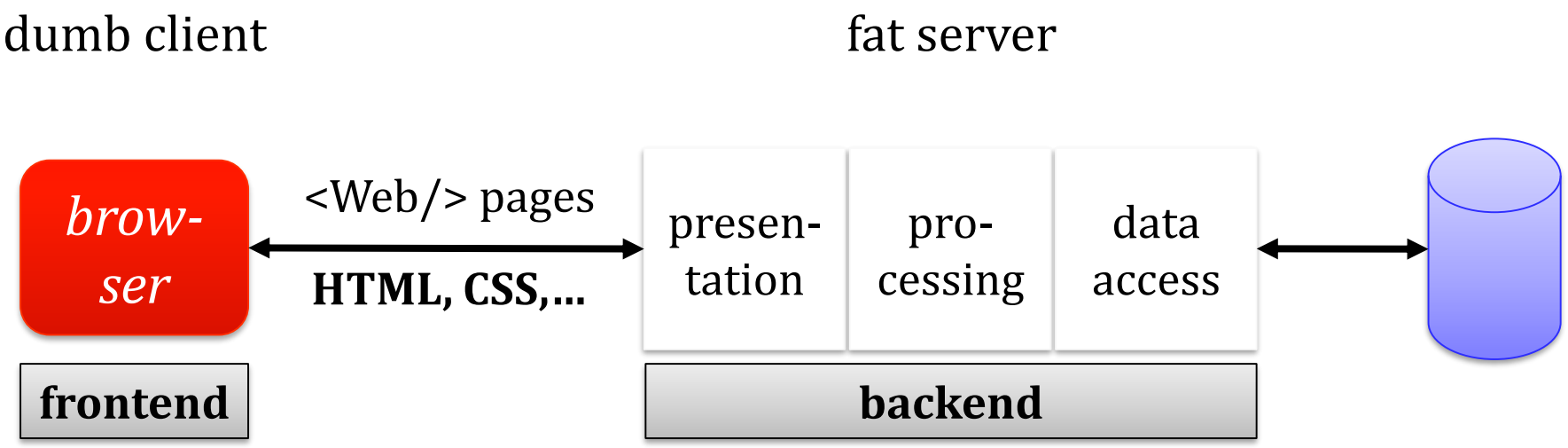


content generation at Web server level

usually, **monolithic application**

www.leaseweblabs.com/2013/10/api-first-architecture-fat-vs-thin-server-debate/

Web application architecture: **classic MV* approach**



design principle:

layers of isolation

changes made in a particular layer have no impact on or do not affect the components in another layer

advanced

implementation

Framework

facilitates complex Web application development, by simplifying usual operations (*e.g.*, access to databases, caching, code generation, session management, access control) and/or encouraging the source code reuse

advanced

implementation

Various frameworks facilitating Web application development on the backend:

ASP.NET: ASP.NET Core MVC
Java: Play, Spring, Sling, Struts, Tapestry, Wicket
JavaScript (Node.js): Express, Koa, LoopBack, Meteor
Perl: Catalyst, Dancer, Mojolicious
Python: CherryPy, Flask, Pylons Project, web2py
Ruby: Padrino, Rails, Sinatra

implementation

Web library

a collection of reusable computational resources
– *i.e.*, data structures + code –
providing specific functionalities (behaviors)
implemented in a given programming language

implementation

Web library

a collection of reusable computational resources
– *i.e.*, data structures + code –
providing specific functionalities (behaviors)
implemented in a given programming language

could be referred by other source code (software):
application server, framework, library,
service, API, or Web component

Open source libraries – examples:

Apache PDFBox – **pdfbox.apache.org**
Beautiful Soup – **www.crummy.com/software/BeautifulSoup**
D3.js – **d3js.org**
Expat – **libexpat.github.io**
ImageMagick – **www.imagemagick.org**
libcurl – **curl.haxx.se**
OpenCV – **opencv.org**
Requests-HTML – **github.com/kennethreitz/requests-html**
SQLAPI++ – **www.sqlapi.com**
TensorFlow.js – **www.tensorflow.org/js/**
zlib – **www.zlib.net**

implementation

Web service

software – remotely used by other applications/services – providing a specific functionality,
frequently through an API (Application Programming Interface)

implementation

Web service

software – remotely used by other applications/services – providing a specific functionality,
frequently through an API (Application Programming Interface)

its implementation must not be known
by the programmer that invokes the service

advanced

implementation

SDK (**Software Development Kit**)

encapsulates the API functionalities into a library
(implemented in a programming language,
for a specific software/hardware platform)

API façade pattern

advanced

implementation

SDK (**Software Development Kit**)

example:
Octokit (for .NET, Objective-C, Ruby,…)
provided by Github – **developer.github.com/v3/libraries/**

advanced

implementation

Web component

part of a Web application
that includes a set of related features

e.g., calendar, news feed reader, URL sharing button

advanced

implementation

Web component

development based on a suite of technologies implemented by the browser – Web Components
developer.mozilla.org/docs/Web/Web_Components

eventually, a JavaScript library or framework can be used, like hybrids, Lit, StencilJS

implementation

Widget

application – stand-alone or included into a container (*e.g.*, an HTML document) – offering a specific functionality

runs on a client (a platform provided by an operating system and/or a Web browser)

implementation

(Web) app

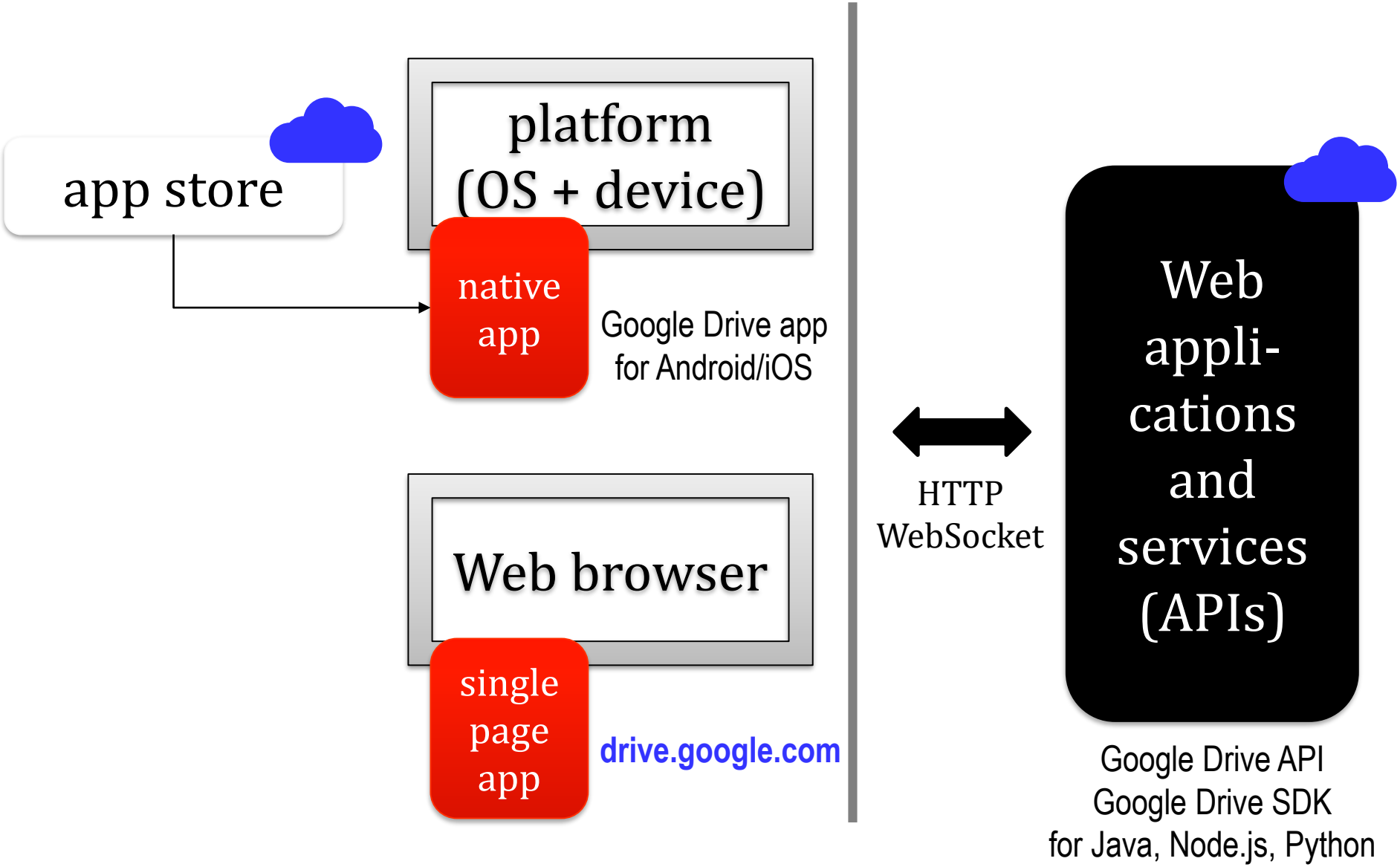
an installable (Web) application
that uses the APIs provided by a platform:
browser, application server, operating system, device,…

implementation

(Web) app

a distributed computer software application designed for
optimal use on specific screen sizes *and*
with particular interface technologies
Robert Shilston, 2013

advanced



adapted from Adrian Colyer (2012)

implementation

Add-on

a generic name for browser-associated applications
(extensions, visual themes, dictionaries,
Web search engine interfaces, plug-ins, etc.)

examples: addons.mozilla.org

development

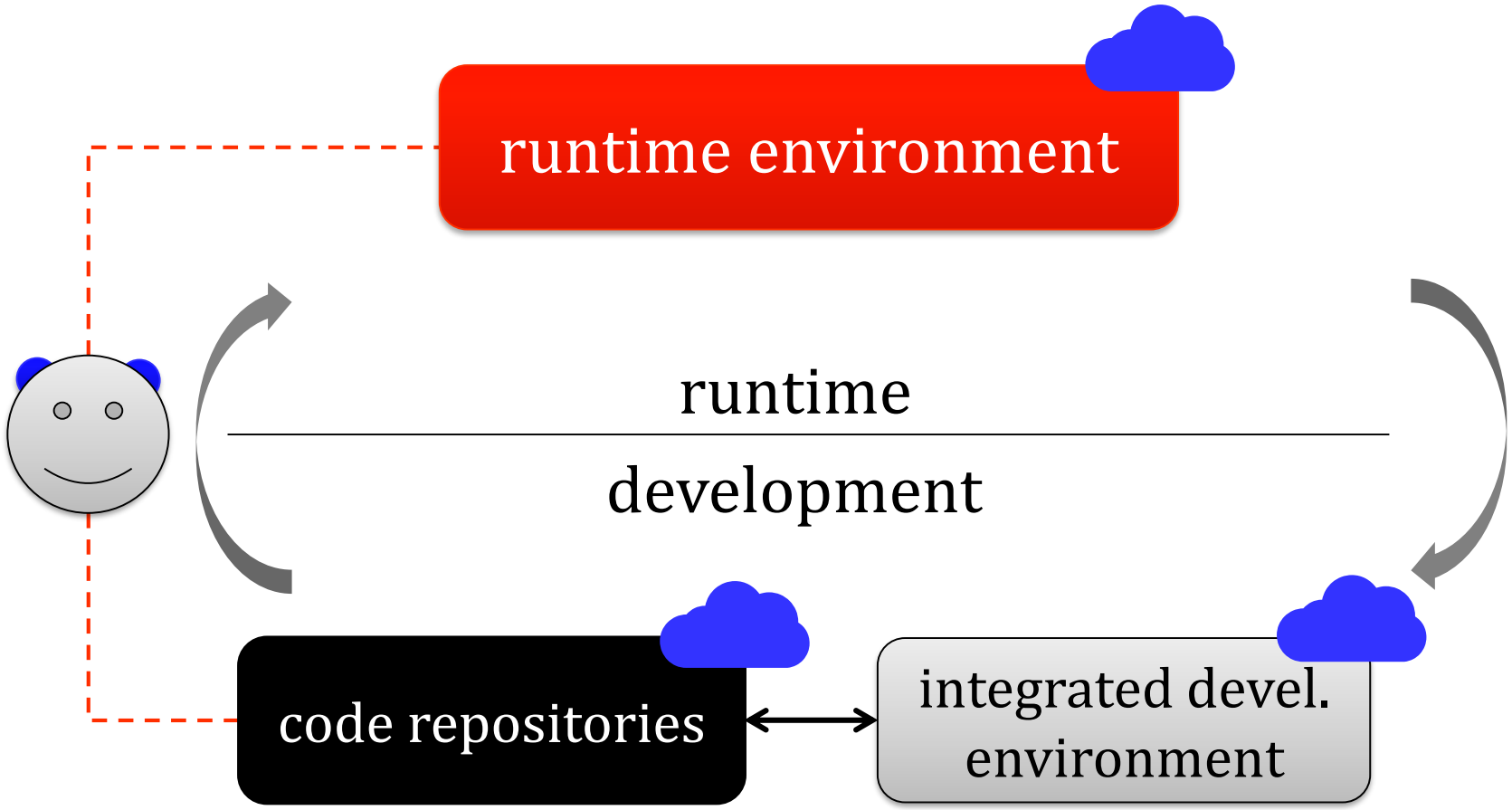
Using development environments

examples – native applications (for desktop):
Anjuta, Aptana Studio, Eclipse, Emacs, IntelliJ IDEA, KomodoIDE, Padre, PHPStorm, PyCharm, RubyMine, Visual Studio (Code), Webstorm

cloud computing-based solutions:
AWS Cloud9, Codenvy, Koding, RELP.it etc.

S. Buraga, *Code in "clouds"*: www.slideshare.net/busaco/cu-codul-n-nori

advanced

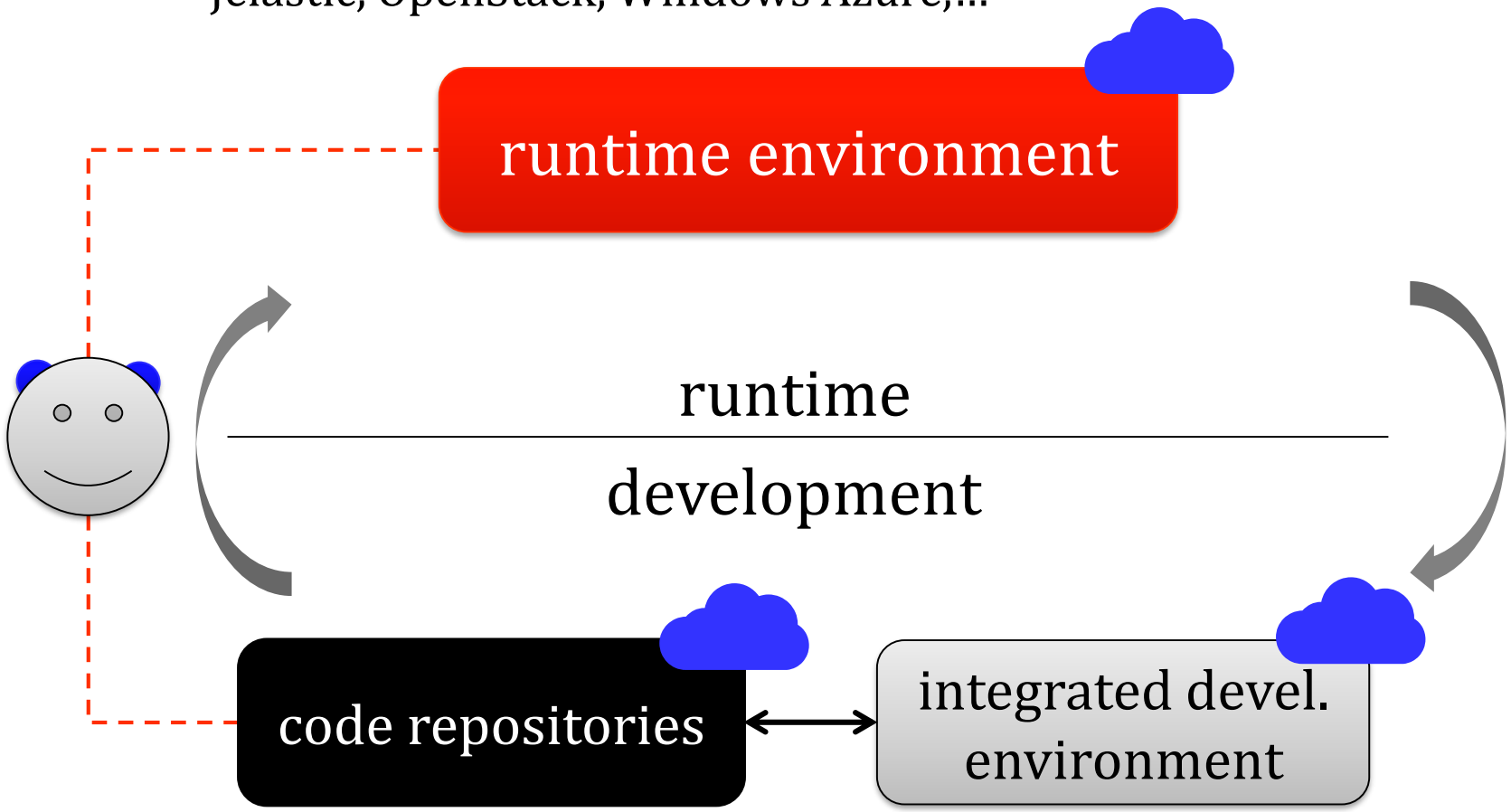


A. Iqbal, M. Haunsenblas, S. Decker (2012)

Development as a Service

advanced

DigitalOcean, Google Cloud Platform, Heroku,
Jelastic, OpenStack, Windows Azure,…



BitBucket,
GitHub

Web: AWS Cloud9, Koding, Ideone, etc.
desktop: Eclipse, Visual Studio (Code),…

useful tools at **github.com/ripienaar/free-for-dev**

development

Automatic generation of documentation,
in various formats

specific instruments: documentation generators

examples:
Doc, Document! X, Doxygen,
JavaDoc, JSDoc, phpDocumentor

development

VCS – Version Control Systems

code review, revision control, versioning

monitoring changes in the source code
made by a team of programmers
on the same group of programs (codebase)

Encouraging/enforcing coding conventions and styles

client level:

HTML + CSS

www.oreilly.com/web-platform/free/files/little-book-html-css-coding-guidelines.pdf

JavaScript – profs.info.uaic.ro/~busaco/teach/courses/staw/web-film.html#week7

server level:

C# – github.com/dennisdoomen/csharpguidelines

Perl – perldoc.perl.org/perlstyle.html

PHP – www.php-fig.org/psr/psr-2/

Python – www.python.org/dev/peps/

Ruby – github.com/styleguide/ruby

Scala – docs.scala-lang.org/style/

for others, see google.github.io/styleguide/

development

Software package management

searching, installing, compiling, dependence checking

examples:
Bower, Composer, npm, NuGet, RubyGems, Yarn

also, study **github.com/showcases/package-managers**

development

Support for workflows
optionally, automatically performed

"producing" a Web application from source code + additional components (build tool)

examples:
Ant, Grunt, Gulp, make, Mimoza, Rake, tup, Yeoman

testing

Tests concerning the source-code

automatic test units – general framework: xUnit
HttpUnit, JUnit (Java), PHPUnit, xUnit.net (C#, F#),
Test::Class (Perl), unittest (Python), Unit.js
+
JSUnit, FireUnit, Mocha, Selenium etc.
for client-side

for details, study **xunitpatterns.com**

testing

Web application specific tests

regarding **content** – structure, HTML + CSS validation,...

hypertext issues (*e.g.*, broken links)

usability – including accessibility, multi-lingual data

Web interface esthetics – hard to evaluate/test

testing

Web application specific tests

component **integration**

permanent **availability** and flexibility
(continuous evolution)

device **independence** – multi-screen
(a large number of possible devices + characteristics)

advanced

testing

Other kinds of testing:

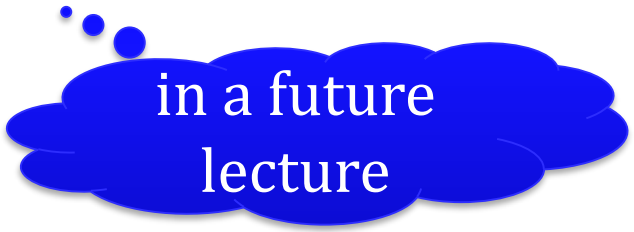
concerning **performance**
load, stressing, continuous testing, scalability

real-life case studies:
High Scalability – **highscalability.com**
Performance Planet – **calendar.perfplanet.com**
Web Performance Stats – **wpostats.com**

testing

Other kinds of testing:

regarding **security**



testing: example

HTML documents – validator.w3.org service
CSS stylesheets – CSS Lint, StyleLint
JSON data – validation using JSONLint
XML documents – well-formed / valid
Client-side scripting (JavaScript) via JS/ES Hint
Server-side (backend) programs – *e.g.*, xUnit
File system integrity and access policies
Database integrity and access policies
Web navigator support – caniuse.com
Security problems – www.owasp.org
Aspects regarding Web application performance

running

Site publishing

dedicated server

vs.

Web hosting provider

free vs. commercial solution

response time, scalability, security, technical support,...

running

Content maintenance (administration)

obtaining, creating, preparing, managing, presenting, processing, publishing, and reusing the content in a systematic and structured manner

advanced

running: management

At the organizational level:
knowledge management

CRM – Client Relationship Management

ERP – Enterprise Resource Planning

workflow management + business rules

EAI – Enterprise Application Integration

advanced

running: management

At the technical level:

managing the content by non-technical personnel
on the basis of separation of concerns principle

Content Management Systems (CMS)

collaborative tools
(*e.g.*, an enterprise wiki)

advanced

running: management

Regarding the user:

Web interaction – *e.g.*, usability
profs.info.uaic.ro/~busaco/teach/courses/hci/

social Web application design patterns
profs.info.uaic.ro/~busaco/teach/courses/hci/hci-film.html#week7

frontend performance
profs.info.uaic.ro/~busaco/teach/courses/staw/web-film.html#week13

advanced

running: usage analysis

explicit methods

based on data provided by users

e.g., surveys and monitoring (user testing, RUM – Real User Monitoring),
e-mail message analysis, reactions on social networks etc.

advanced

running: usage analysis

implicit methods

automatic data collecting (user analytics)

usually, via cookies

advanced

running: usage analysis

user profile generation: Web usage mining

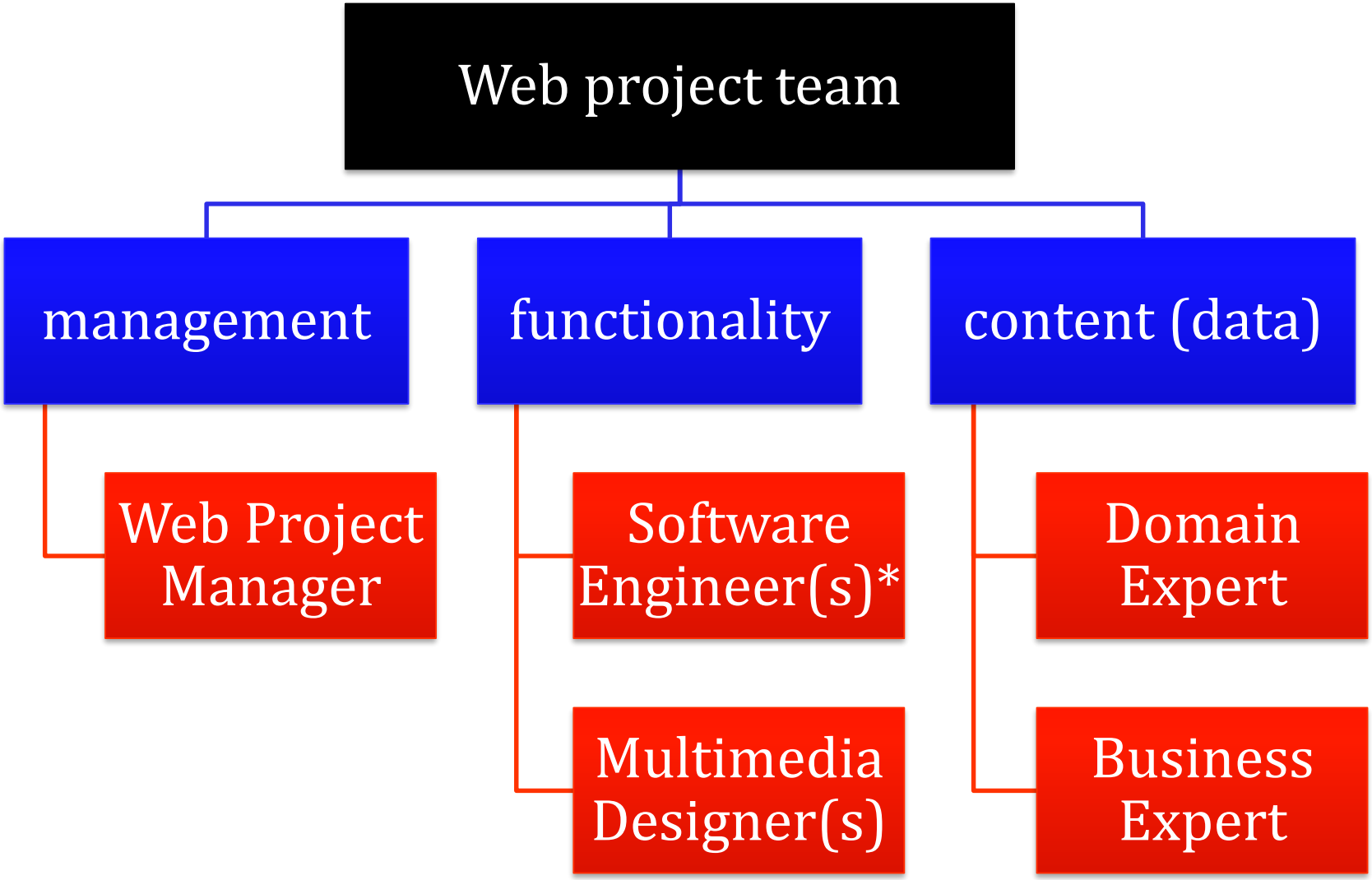
log file analysis
(*e.g.*, access.log – Apache, AWStats,...)

Website "popularity" measurement:
load speed, number of accesses, visit time + duration etc.

monitoring/reporting services
examples: Matomo, Plausible.io, WordPress Statistics

parameters of a web project

main objective
duration
cost
methodology
technologies
processes
outcome
human resources
team profile



*__frontend__ or __backend__ or __full-stack__ (frontend & backend)

www.slideshare.net/busaco/sabin-buraga-dezvoltator-web-n-2019

Several examples regarding
Web application architectures?

case study: flickr

Aim:
online sharing of graphical content (photos)

representative social Web application

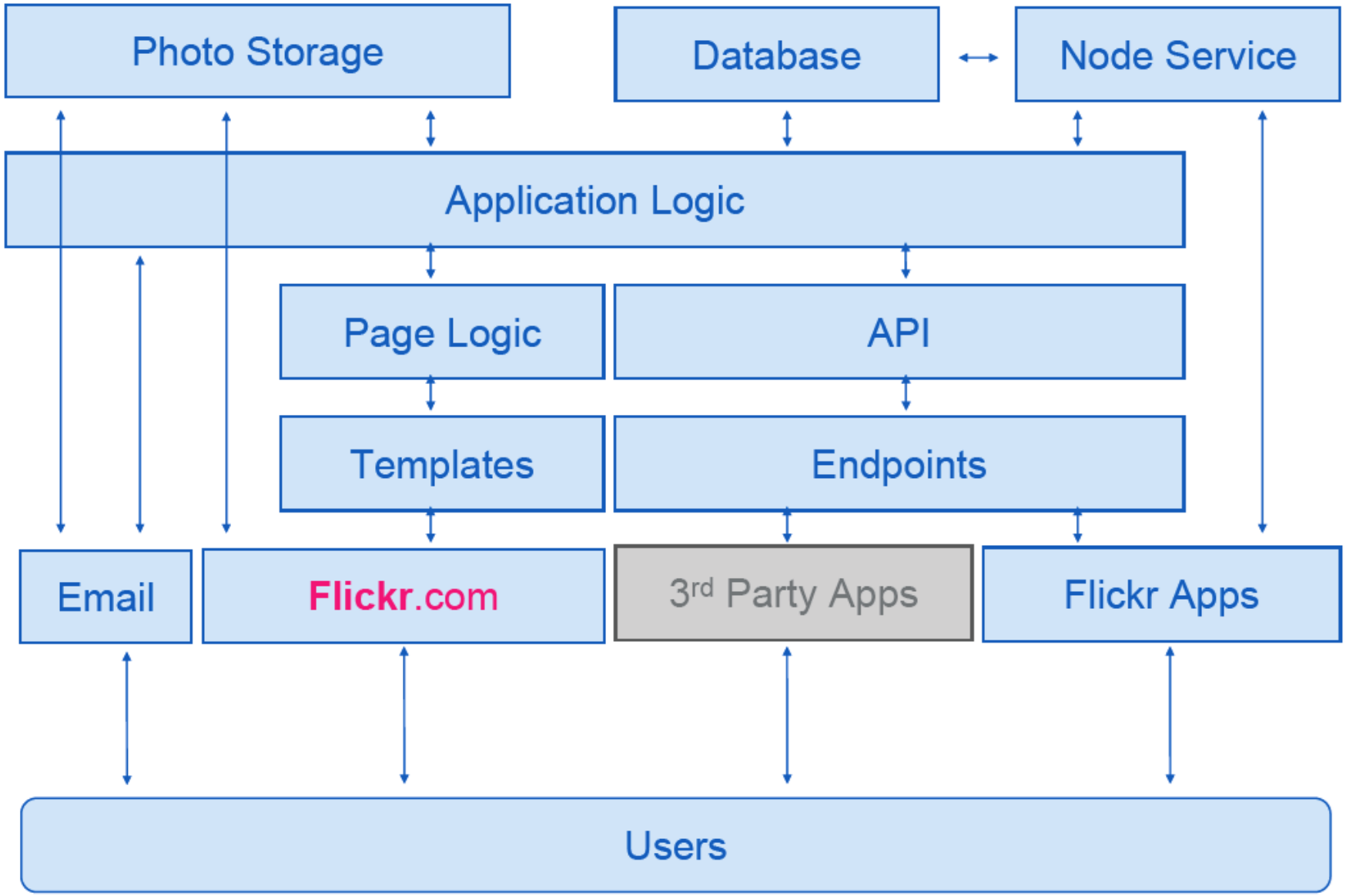
community aggregation – image as social object

support for annotations via tagging
+ comments

case study: flickr – technologies

PHP (processing – application logic, access to API, content presentation via Smarty, e-mail module)
Perl (data validation)
Java (storage node management)
MySQL (data stored in InnoDB format)
ImageMagick (C library for image processing)
Ajax (asynchronous interaction)
Linux (runtime platform)

other details at **highscalability.com/flickr-architecture**



initial architecture – according to Cal Henderson, 2007

API Kits

advanced

C
- Flickcurl

Cold Fusion
- CFlickr

Common Lisp
- Clickr

cUrl
- Curlr

Delphi
- dFlickr

Go
- go-flickr

Java
- Flickr4Java
- flickr-jandroid

.NET
- Flickr.NET

Node.js
- node-flickrapi

Objective-C
- ObjectiveFlickr
- FlickrKit

Perl

application programming interfaces (APIs) provided by Flickr

facilitate the access to Web services for applications running on various platforms

requests: REST, XML-RPC, SOAP
responses: REST, XML-RPC, SOAP, JSON

www.flickr.com/services/api/

generic aspects concerning the system design:

resource categories: *user* + *picture*

relations between *user* instances (e.g., *follow*)

relations between *user* and *picture* instances
(*make*, *depicts*, *comment*, *like*,…)

performance features:
response time, scalable software architecture,
scalable persistent storage, image optimization

recommending resources (*user*/*picture*) of interest

details in *Create a Photo Sharing App* (2016)
blog.gainlo.co/index.php/2016/03/01/system-design-interview-question-create-a-photo-sharing-app/

case study: netflix

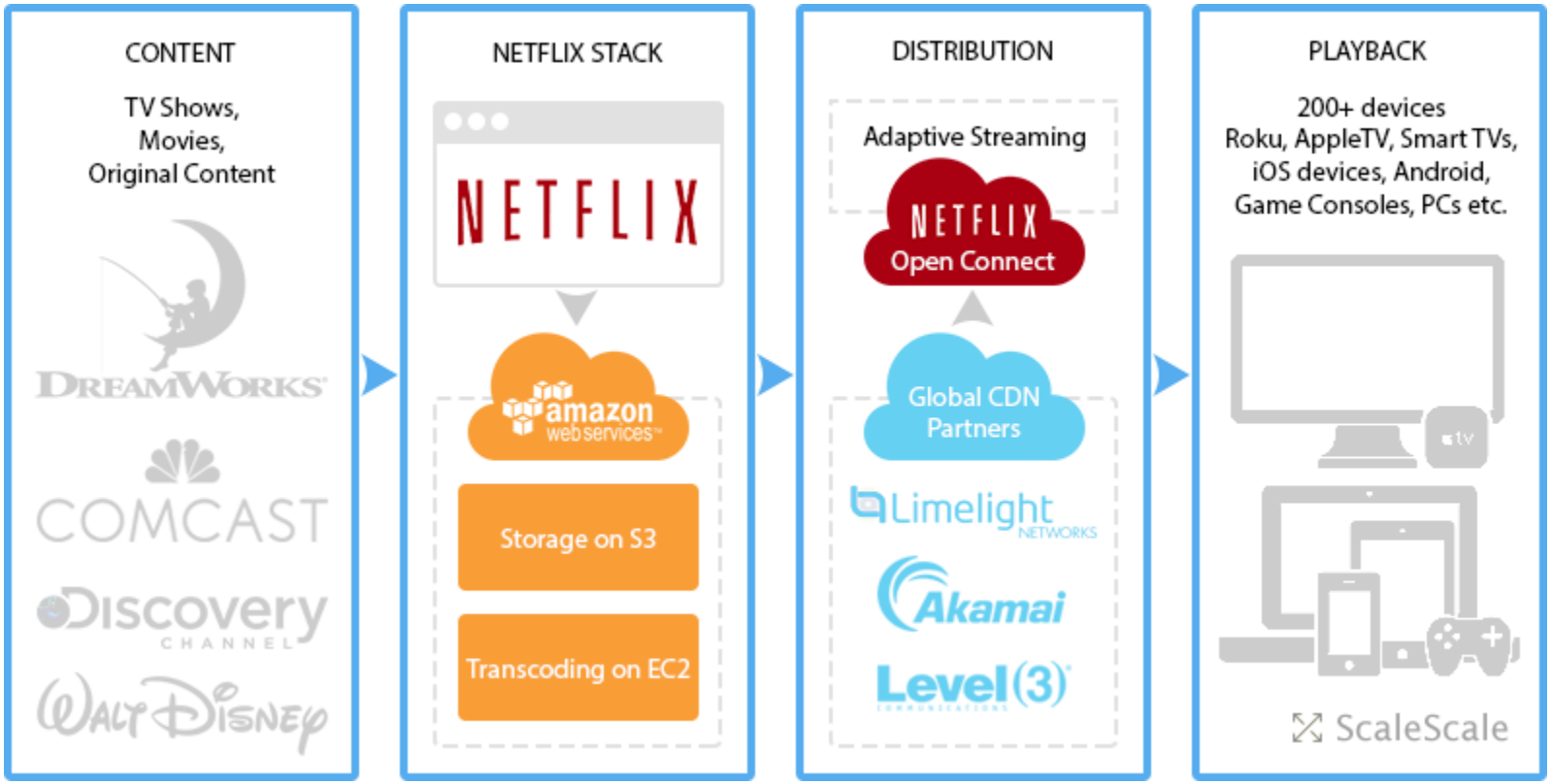
Aim: providing video content on demand (streaming) + Web TV

services available on multiple devices/platforms

cloud-based deployment

adopting, among others, various open technologies

netflixtechblog.com



aspects of interest:
content (movies, TV shows,…),
storage & encoding/decoding (transcoding),
adaptive streaming,
playback (support for heterogeneous devices)

backend processing	Flask (Python), Java, Node.js
frontend processing	React, WinJS (JavaScript)
storage systems	MySQL, PostgreSQL, Oracle, AtlasDB, Cassandra, Hadoop, Presto etc.
cloud services	Amazon EC2 (video processing) Amazon S3 (storage)
SQL services	Amazon RDS (Relational DB Service)
NoSQL services	Amazon DynamoDB
code management	GitHub (implemented in Ruby + C)
continuous integration	Jenkins (Java implementation)
server management	Apache Mesos (written in C++)
content distribution networks	Open Connect CDN (FreeBSD, Nginx), Akamai, Level 3, Limelight
monitoring	Boundary, LogicMonitor, Vector,…

highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html
stackshare.io/netflix/netflix

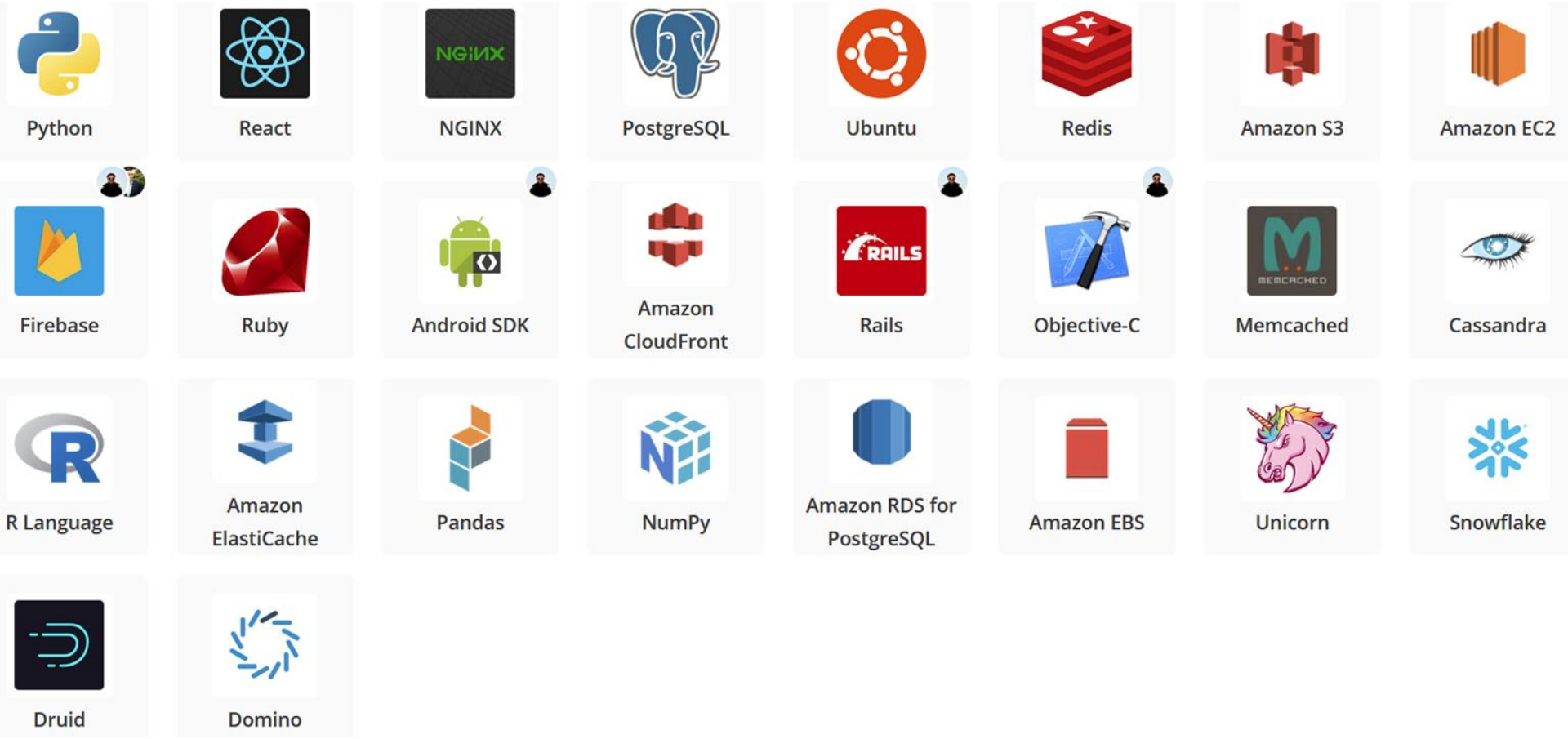
case study: instacart

Purpose: multi-platform Web application offering fast home delivery services for grocery products

products: 500 millions
local shops (offerers): 40 de mii
localitites: 5500 (US + Canada)
orders: millions/year

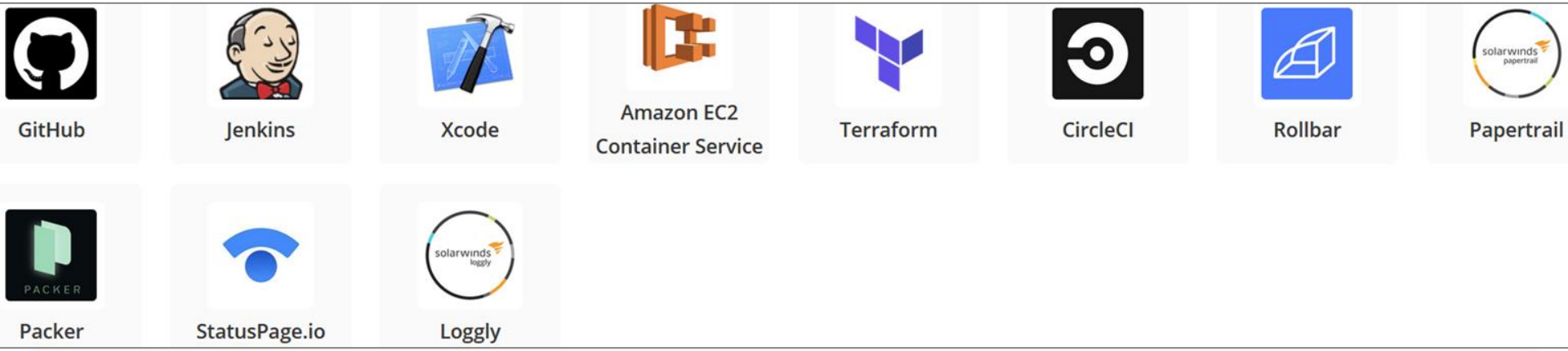
stackshare.io/instacart/instacart/

case study: instacart



technologies + main tools – application & data

additional instruments, tools – utilities



tools for development operations – DevOps

internal processes regarding the business – business tools

Web application	Backend processing	Persistent storage
Amazon	Perl, Java	MySQL (MariaDB), Amazon DynamoDB, Amazon SimpleDB, Amazon ElastiCache
Coursera	Django (Python), Node.js (JavaScript), Play (Scala)	MySQL, Apache Cassandra
DuckDuck Go	Node.js	PostgreSQL
Facebook	Hack, PHP (HHVM), Tornado (Python), Java, JavaScript	RocksDB, Presto, Cassandra, Beringei
Forbes	ASP.NET, Node.js, PHP 7, Ruby	MySQL (MariaDB)
Google	C++, Dart, Go, Java, Python	BigTable, MariaDB

stackshare.io/stacks

Web application	Backend processing	Persistent storage
Linkedin	Grails (Java), JavaScript, Scala	MySQL, Oracle DB, RocksDB, Hadoop
Lyft	PHP, Flask (Python), Java, Go, C++	MongoDB, DynamoDB, Redis
Medium	Node.js, Go	Neo4j, DynamoDB, Redis
Pinterest	Django (Python), Java, Go	MySQL, Hadoop, Apache HBase, Redis, Memcached
Stack Overflow	.NET Framework (C#)	MS SQL Server, Redis
Sound Cloud	Clojure, Scala, JRuby	MySQL

stackshare.io/stacks

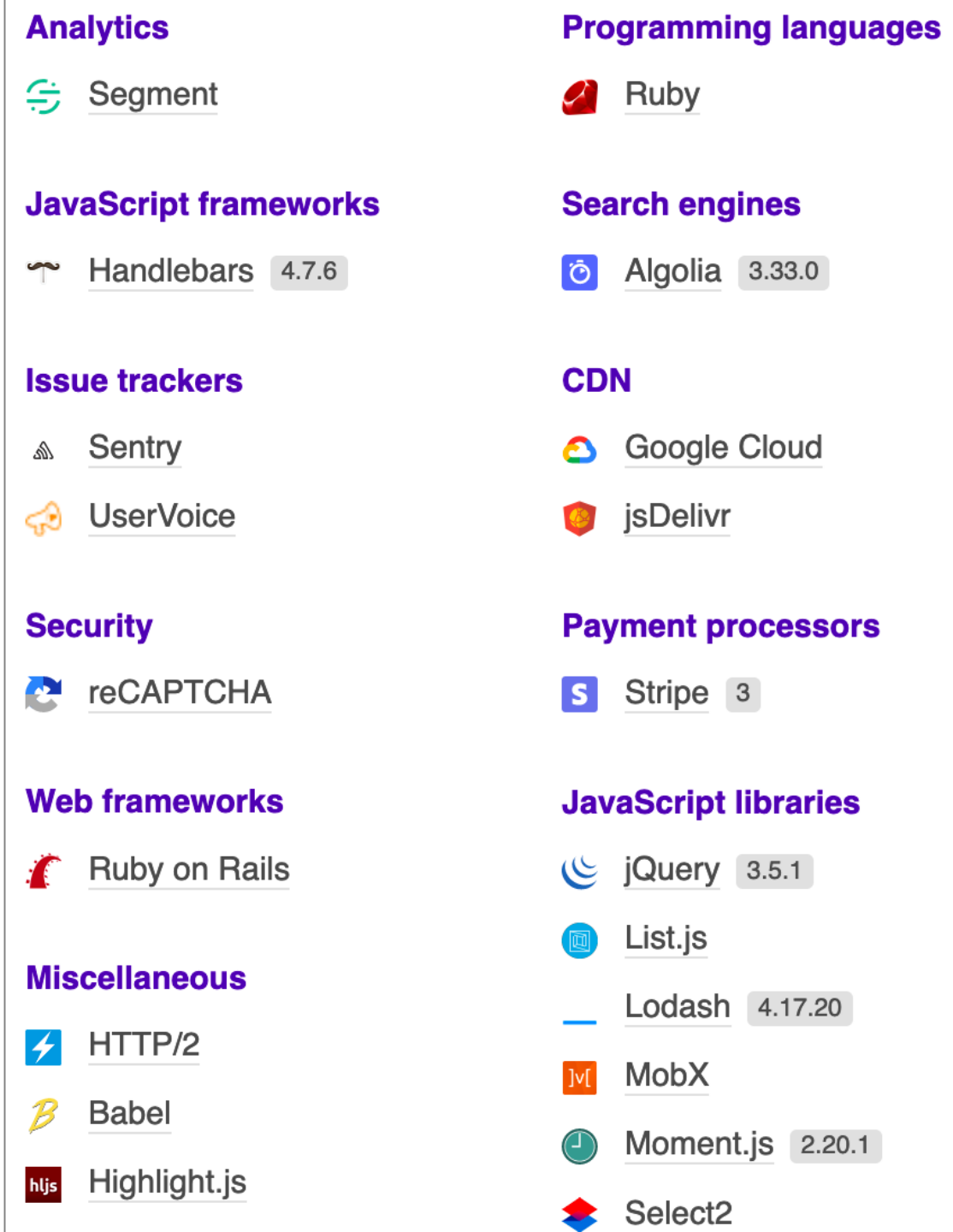
Inspecting the technologies used by a Web application with the instruments

Wappalyzer

www.wappalyzer.com

WhatRuns

www.whatruns.com



How is an application server used for developing a Web application?

web application server

Purpose:
increasing the efficiency of all processes involved in the development of Web applications

web application server

Integrated in one/many Web servers

can also provide its own Web server
or runtime environment

web application server

Can encourage or enforce an architectural vision regarding Web application development

typical situation:
MVC or variants

web application server

Simplifies the invocation of programs (scripts) concerning a Web application

- dynamic content generation on the server side

web application server

Aspects of interest:

programming language(s)
core API
data models persistent storage
Web interaction
cookies and sessions
development environments + frameworks, components,…
specific characteristics

web application server

Programming language(s)

static – example: C#, Java, Rust
versus
dynamic – *e.g.*, JavaScript, PHP, Python, Ruby

web application server

Programming language(s)

interpreted (Perl, Python, Ruby)
and/or compiled (C, C++, Rust)

commonly, intermediate code generation is preferred:
IL (Intermediate Language) – C#, Erlang, Java, Scala,…

web application server

Programming language(s)

interpreted (Perl, Python, Ruby)
and/or compiled (C, C++, Rust)

commonly, intermediate code generation is preferred:
IL (Intermediate Language) – C#, Erlang, Java, Scala,…

more recently, at runtime: Just-in-Time (JIT)
in the Web context: JavaScript, PHP 8, TypeScript

web application server

Core API

gives the language + application server “power”
(via predefined functions/classes)

web application server

Core API

gives the language + application server “power”
(via predefined functions/classes)

security, consistency, access to the operating system/runtime resources, platform independence etc.

web application server

Persistent storage

in relational databases – using SQL

web application server

Persistent storage

in relational databases – using SQL

examples:
ADO.NET for ASP.NET
Java – JDBC (Java DataBase Connectivity)
PHP – predefined functions/modules, plus included libraries (SQLite + mysqli) or various extensions

advanced

web application server

Persistent storage

in relational databases – using SQL

ORM (Object-Relational Mapping): Data Mapper pattern

Go – Xorm framework
Java – JPA (Java Persistence API) specification
+ implementations: EclipseLink, Hibernate, OpenJPA,…
Node.js – Sequelize library
PHP – frameworks: Cycle ORM, Doctrine, Propel, RedBean etc.

advanced

web application server

Persistent storage

in relational databases – using SQL

Active Record - architectural pattern used in ORM to encapsulate a table or a view into a class

examples:
active_record, TypeORM (Node.js modules), Castle Project (.NET), DBIx::Class (Perl), Django ORM and Orator (Python), Play Framework (Java, Scala), Rails (Ruby)

web application server

Persistent storage

tree-based models: XML

(semi)structured data
transformations in other formats: XPath, XSLT
processing: DOM, SAX, SimpleXML etc.
data validation: DTD, XML Schema, RELAX,…
querying: XQuery

advanced

web application server

Persistent storage

using other non-relational paradigms
(based on graphs and/or key-value)

distributed on Internet, scalable – NoSQL

github.com/erictleung/awesome-nosql

examples:

Cassandra, MarkLogic, MongoDB, Neo4j, OpenLink Virtuoso, Redis

web application server

Web interaction

facilitated by specific controls specified in the source code invoked on the server

web application server

Web interaction

facilitated by specific controls specified in the source code invoked on the server

can emulate HTML form fields and/or provide new interactive controls – *e.g.*, calendar, slideshow,…

- generation of processable code at client level (front-end)
  Web components (HTML + CSS + JavaScript) executed by the browser

advanced

web application server

Web interaction

examples:

ASP.NET (**<asp:control>** – *e.g.*, FileUpload, ListBox, Table,...)

PRADO framework (PHP)

formidable, form-data, forms – Node.js modules

Java platform: JSF (Jakarta Server Faces)

web application server

Web interaction

encourages the use of templates based on a specific processor

Web template system

web application server

Web interaction

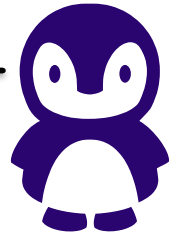
Web template system

using specifications of content presentation
(a Web template), persistent data (*e.g.*, retrieved from
a database) is used by a processor (template engine)
in order to generate HTML documents or other formats

```
<!-- HTML template -->
<h1 class="profile">[@username] profile</h1>
<img src="[@photoURL]" alt="photo"/>
<div class="identity">[@firstName] [@lastName]</div>
<div class="location">[@location]</div>

<!-- PHP program: template processing -->
$profile = new Template ('templates/profile.tpl');
$profile->set ('username') = 'Tux';
$profile->set ('photoURL') = 'imgs/tux.svg';
$profile->set ('firstName') = 'Tuxy';
$profile->set ('lastName') = 'Pinguinesscool';
$profile->set ('location') = 'Romania';
```

Tux profile



Tuxy Pinguinesscool
Romania

example:
the specification of content presentation
(HTML template – **.tpl**) includes variable names
– here, using **[@*variable*]** syntax – that will be substituted by
actual values obtained from the program as a result
of the Web template system invocation

web application server

Web interaction

Web template system

on the server

Apache FreeMarker (Java), Blade (PHP), Haml (Ruby), Mustache (C++, JS, PHP, Python, Scala,…), Pug (Node.js), Razor (.NET), Smarty (PHP), Tonic (PHP), XSLT (XML)

web application server

Web interaction

Web template system

on the client

available for JavaScript:
EJS, HandleBars, Mustache.js, Nunjucks,...
github.com/sorrycc/awesome-javascript#templating-engines

advanced

web application server

Web interaction

cookie and session management

via data structures/types – examples:
HttpSession class (ASP.NET), HttpSession interface (Java servlets), HTTP::Session (Perl), session (Flask – Python framework), web.session (web.py), HttpFoundation (Symfony component – PHP framework), SessionComponent class (CakePHP), session array (Ruby on Rails), play.mvc.Http.Cookie (Play for Java/Scala), sessions (Gorilla – Go) cookie-parser and express-session (Node.js modules for Express)

web application server

Web interaction

asynchronous data transfer via Ajax technology suite

possibly, via additional frameworks/modules/classes

advanced

web application server

Support for software engineering

encouraging/enforcing the use of **design patterns** (arhitectural, behavioral, structural, targeting concurrent and/or distributed processing and others)

AMI (asynchronous method invocation), broker, CBD (components-based development), DAO (data access object), DDD (domain-driven design), DTO (data transfer object), façade, MOM (message-oriented middleware), microservice, **MV***, publish-subscribe, SOA (service-oriented architecture), singleton,…

example: *Spaghetti Code* **anti-pattern**

usually, specific to Web applications that mix business logic with content presentation (view) and data model access mechanism

sourcemaking.com/antipatterns/spaghetti-code

advanced

web application server

Support for software engineering

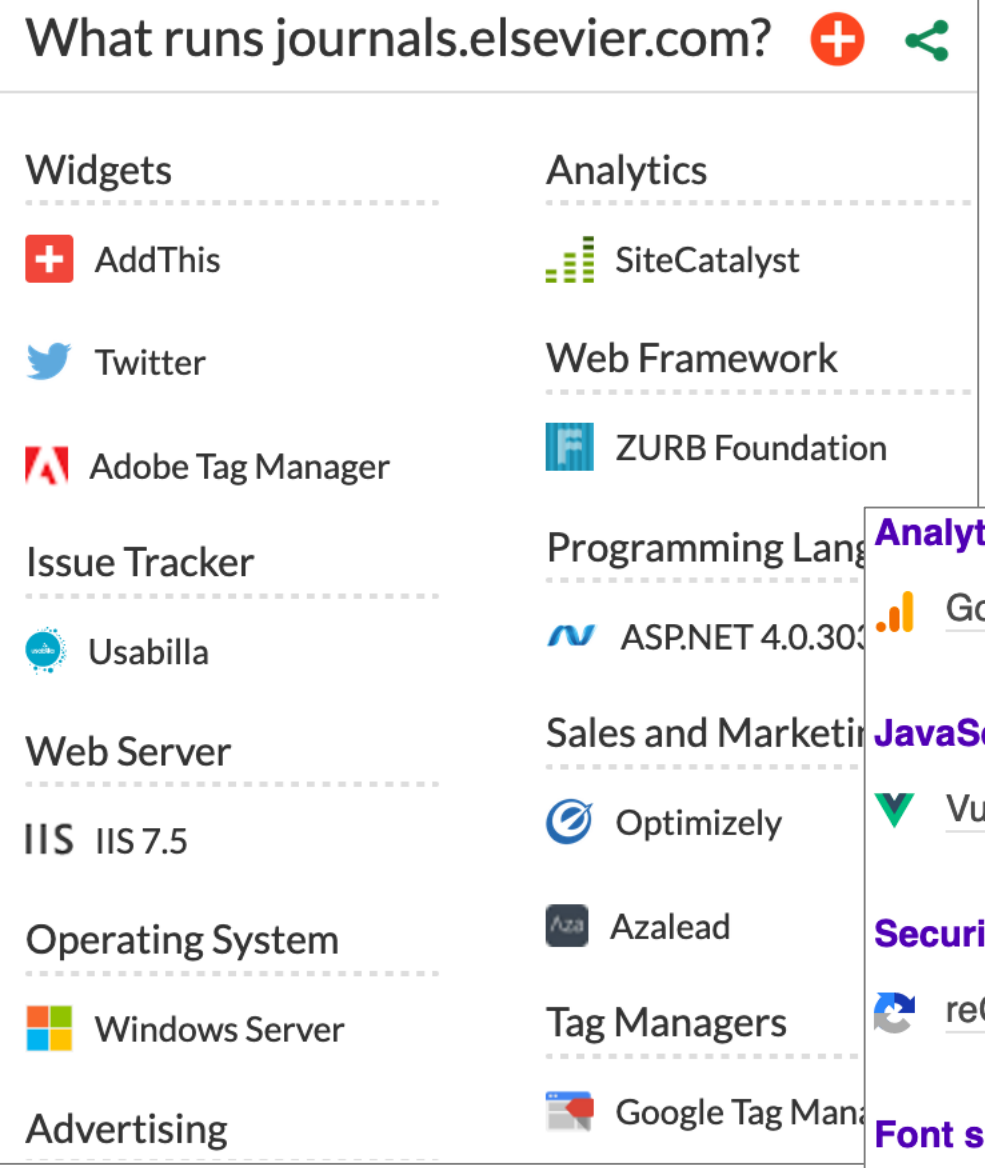
encourages the adoption of an architecture:
layered
event-driven
microkernel / plug-in
microservices
space-based, cloud

according to M. Richards, *Software Architecture Patterns*, O'Reilly, 2015

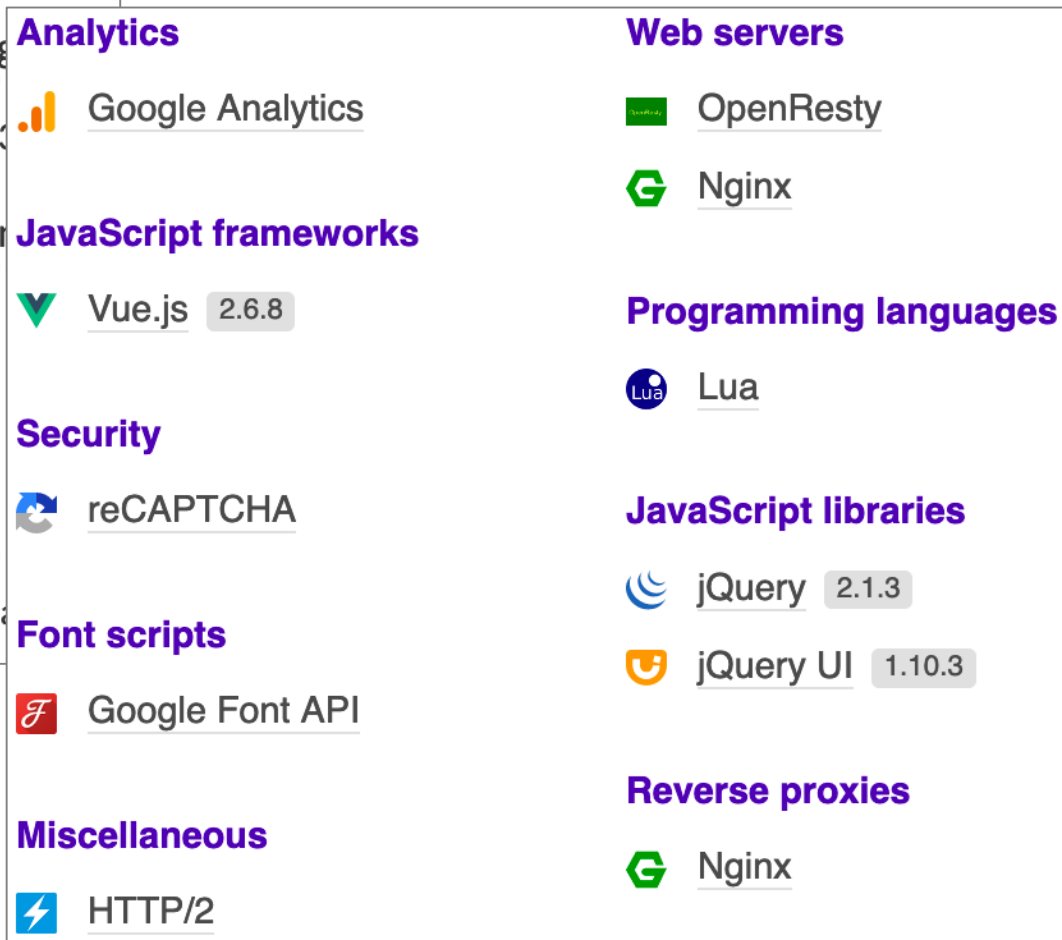
web application server

Examples – including frameworks

C++ – CppCMS, Silicon, TreeFrog, Wt
C# *et al.* (.NET platform) – ASP.NET
D – Dweb, vibe.d
Dart – Aqueduct, shelf, Stream,…
Erlang / Elixir – Chicago Boss, Phoenix, N2O, Sugar
Go – Beego, Buffalo, Gin, Gorilla, Revel
Haskell – Snap, Spock, Yesod etc.



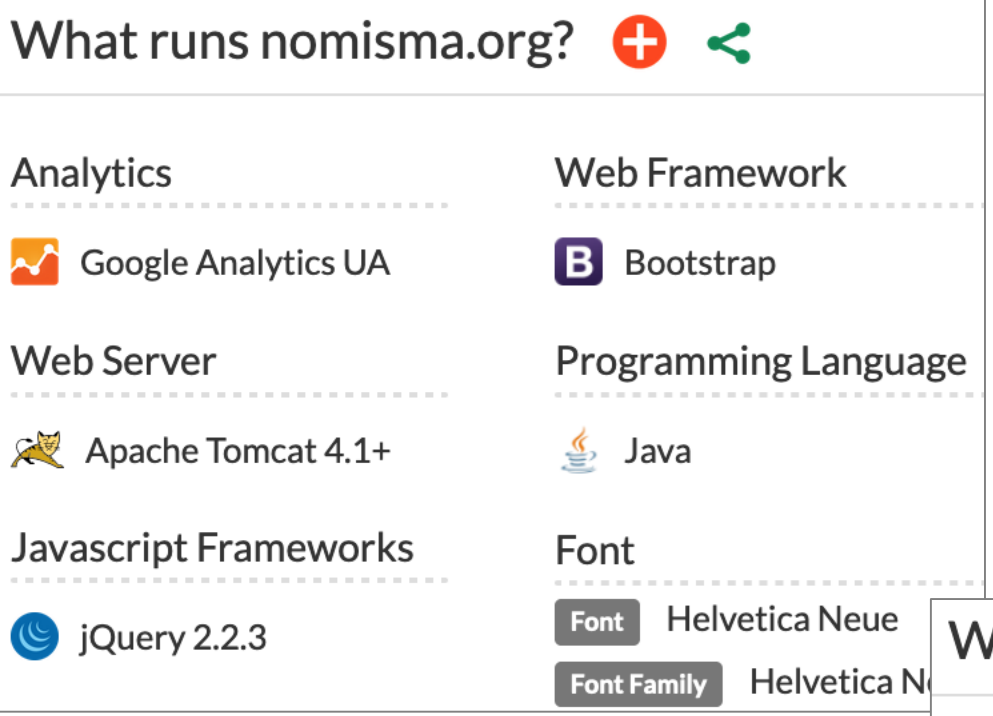
inspecting the technologies used by a Web application with the instruments Wapplyzer and WhatRuns

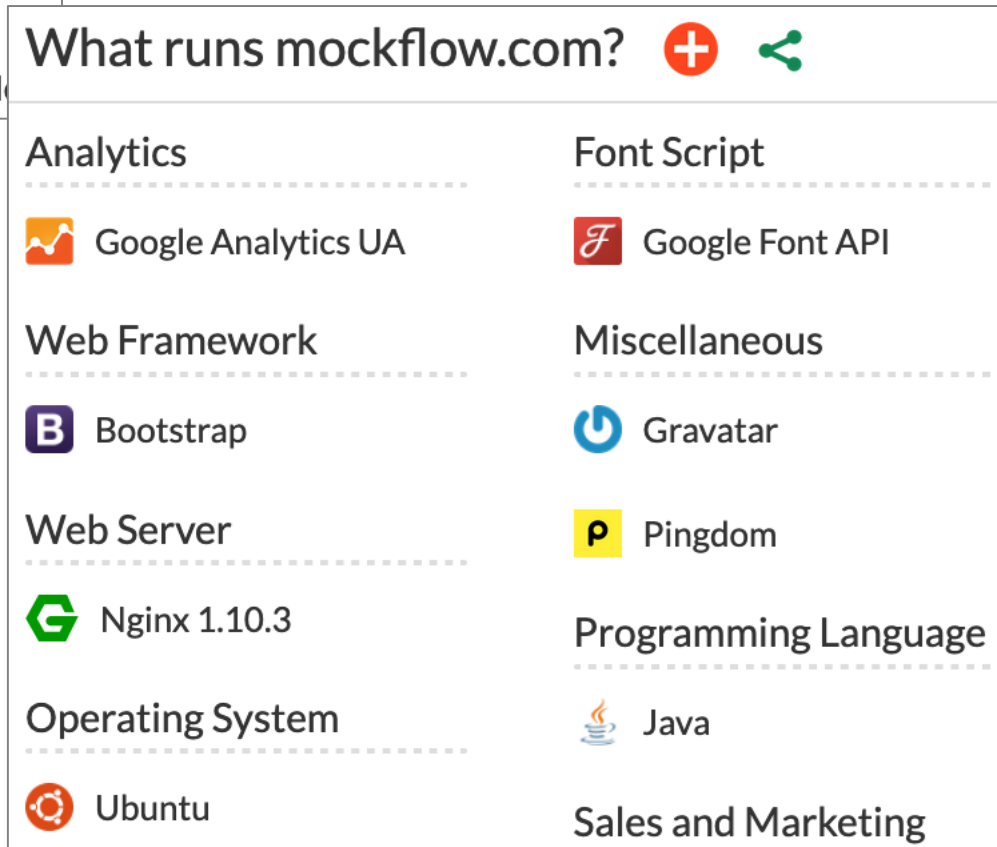


web application server

Examples – including frameworks

Java (Jakarta Enterprise Edition – ex-J2EE) – GlassFish, JBoss, Tomcat, Payara + Apache Struts, Apache Wicket, Grails, Spring, Vaadin
JavaScript (ECMAScript) – Node.js + Derby, Express, Locomotive, Meteor,…
Perl – Catalyst, CGI::Application, Mojolicious
PHP – PHP





inspecting the technologies used by a Web application with the instrument WhatRuns

web application server

Examples – including frameworks

Python – Gunicorn, Tornado, uWSGI, Waitress etc.
(based on WSGI – *Python Web Server Gateway Interface* spec)
+ Django, TurboGears, web2py
Ruby – Passenger, Puma, Thin, Unicorn,…
(implements Rack specification)
+ Ruby on Rails, Sinatra
Rust – Gotham, Iron, Rocket, Rouille etc.
Scheme – Artanis
TypeScript – Deno, LoopBack

What runs httparchive.org?

Analytics
- Google Analytics UA
- SpeedCurve

Font Script
- Font Awesome
- Google Font API

Web Framework
- Bootstrap

Web Server
- gunicorn 19.7.1

Javascript Graphics
- Highcharts

Programming Language
- Python

Font
- Font: Open Sans
- Font Family: Open Sans, San

What runs codepen.io?

Analytics
- Google Analytics UA

Font Script
- Google Font API

Web Framework
- Ruby on Rails

Web Server
- Phusion Passenger

Programming Language
- Ruby

CDN
- CloudFlare

Advertising
- BuySellAds

Tag Managers
- Google Tag Manager

inspecting the technologies used by a Web application with the instrument WhatRuns

web application server

Examples – including frameworks

multi-language approach:
Vert.x
(available for Java, JS, Groovy, Ruby, Ceylon, Scala,…)

web application server

Integrated into a software stack

software stack (servers, tools,…)
offering support for developing complex Web applications

web application server

Integrated into a software stack

software stack (servers, tools,…)
offering support for developing complex Web applications

available – usually, *open source* –
for a specific platform
(operating system, Web server, database server,
application server, programming language)

web application server

Integrated into a software stack

LAMP
(Linux, Apache HTTP Server, MariaDB/MongoDB, Perl/PHP/Python)

alternatives:
FAMP (FreeBSD), MAMP (macOS),
WAMP (Windows), XAMP (multi-platform)

www.apachefriends.org

advanced

- Adminer
- Apache
- Drupal
- Joomla
- Laravel
- Mariadb
- Memcached
- Mongodb
- Mysql
- Nginx
- Php
- Phpmyadmin
- Postgresql
- Redis
- Wildthing
- Wordpress

www.wamp.net

tools for Web developers

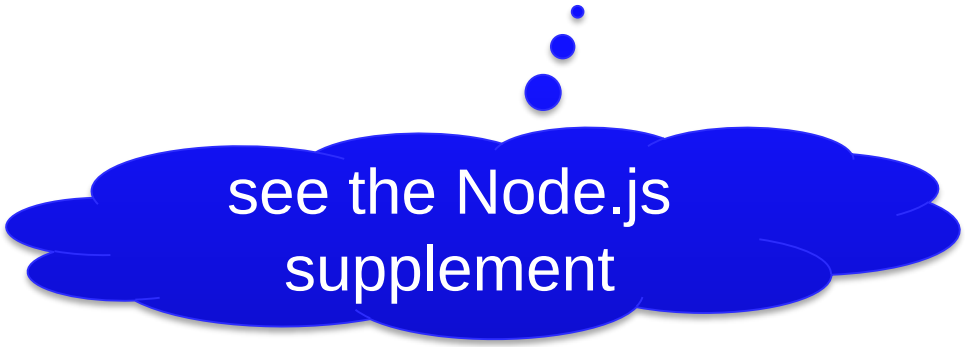
advanced

web application server

Integrated into a software stack

based on JavaScript – full stack Web development

MEAN (MongoDB, Express, Angular, Node.js)
MERN (MongoDB, Express, React, Node.js)



advanced

web application server

Integrated into a software stack

complementary approaches:

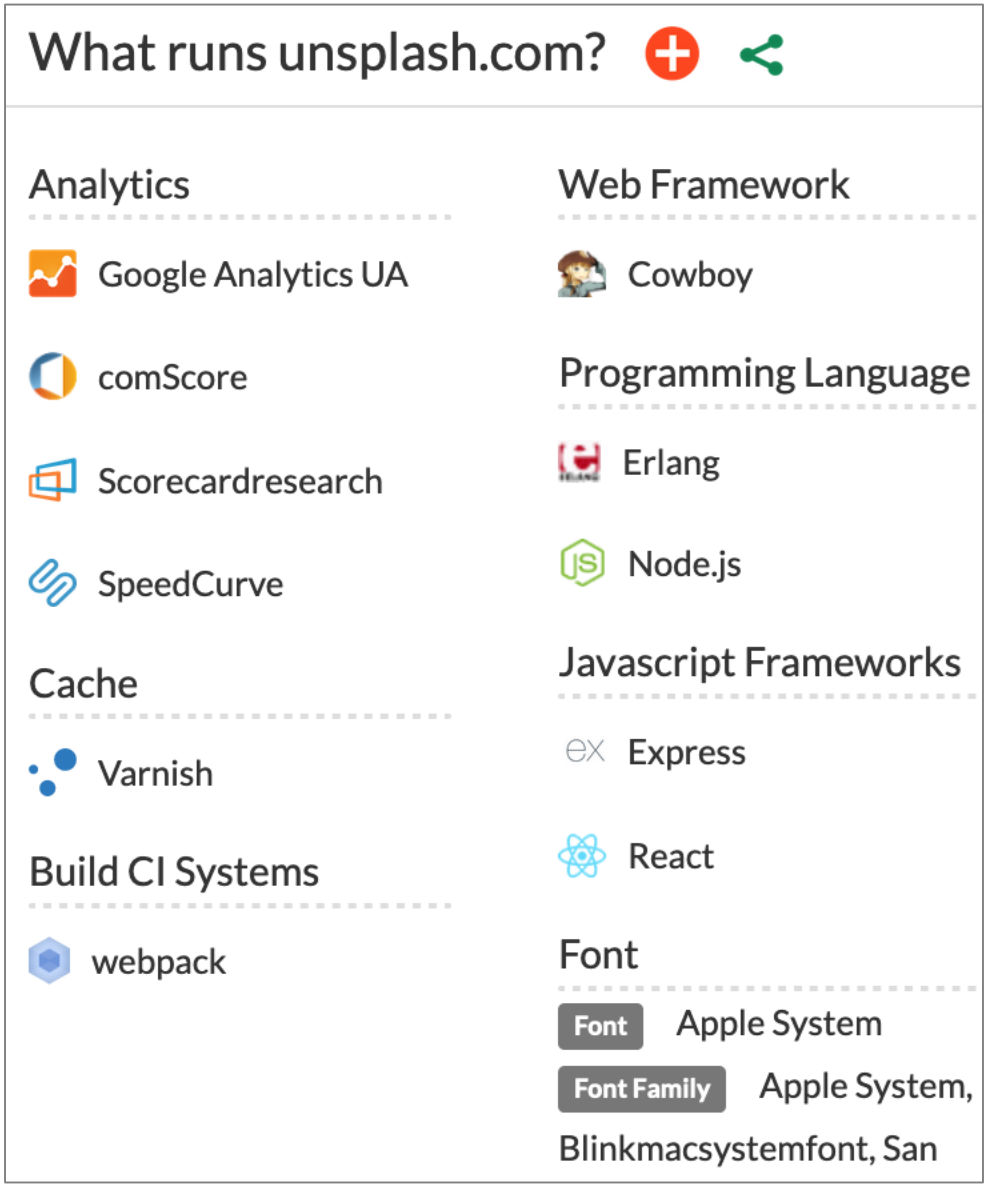
LAPP (Linux, Apache, PostgreSQL, Perl/PHP/Python)

LEMP (Linux, Nginx, MySQL/MariaDB, Perl/PHP/Python)

LLMP (Linux, Lighttpd, MySQL/MariaDB, Perl/PHP/Python)

LYCE (Linux, Yaws, CouchDB, Erlang)

LYME (Linux, Yaws, Mnesia, Erlang)



inspecting the technologies used by a Web application with the instrument WhatRuns

Essential information about PHP?

php

History

Important characteristics

PHP as programming language

paradigms: procedural, object-oriented, functional

PHP as Web development platform

interaction, database access, frameworks, libraries and tools, case studies

Personal Home Page Tools (1995)
Rasmus Lerdorf

PHP 3 (1998)
developed by Zend – Zeev Suraski & Andi Gutmans

PHP 4 (2000)
support for object-oriented programming

PHP 5 (2004) – most recent version: **PHP 5.6** (2014)
new features inspired by Java

PHP 7 (2015), **PHP 7.2** (2017), **PHP 7.4** (2019)
strong typing, support for Unicode, performance,…

PHP 8 (2020), **PHP 8.1** (November 2021)
major update: real-time compilation, new data types and syntactic constructs, improvements etc.

php: characteristics

Web application server

provides a script-based interpreted programming language

can be directly included into HTML documents

php: characteristics

PHP is a procedural language, offering support for other programming paradigms
(object-oriented and, recently, functional)

can be used as a general purpose language, too

php: characteristics

Syntax inspired by C, Perl and Java – case sensitive

whitespaces (space character, Tab, New Line) have no effect on the execution of the program

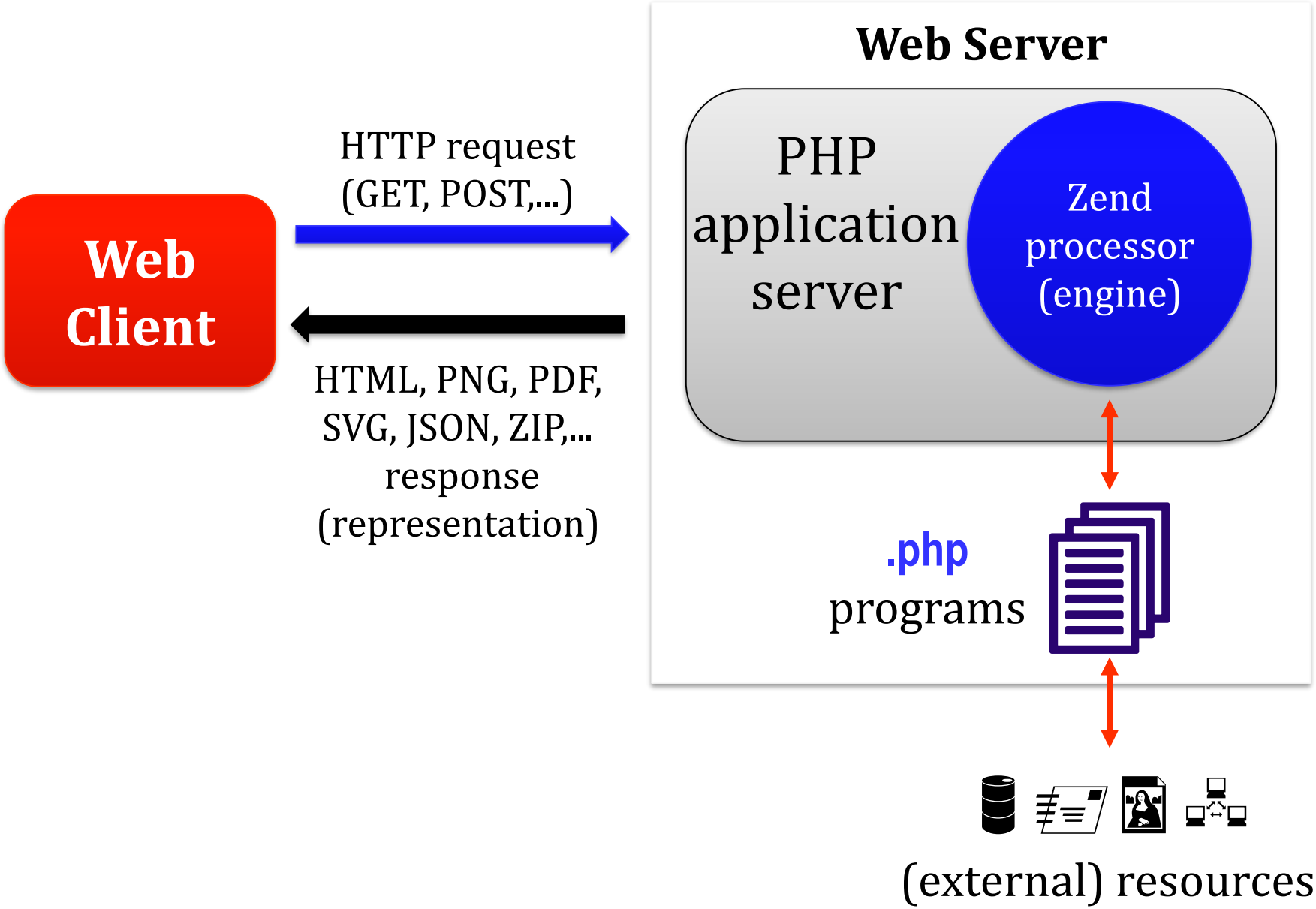
usually, the files which contain PHP source code have the extension **.php**

php: characteristics

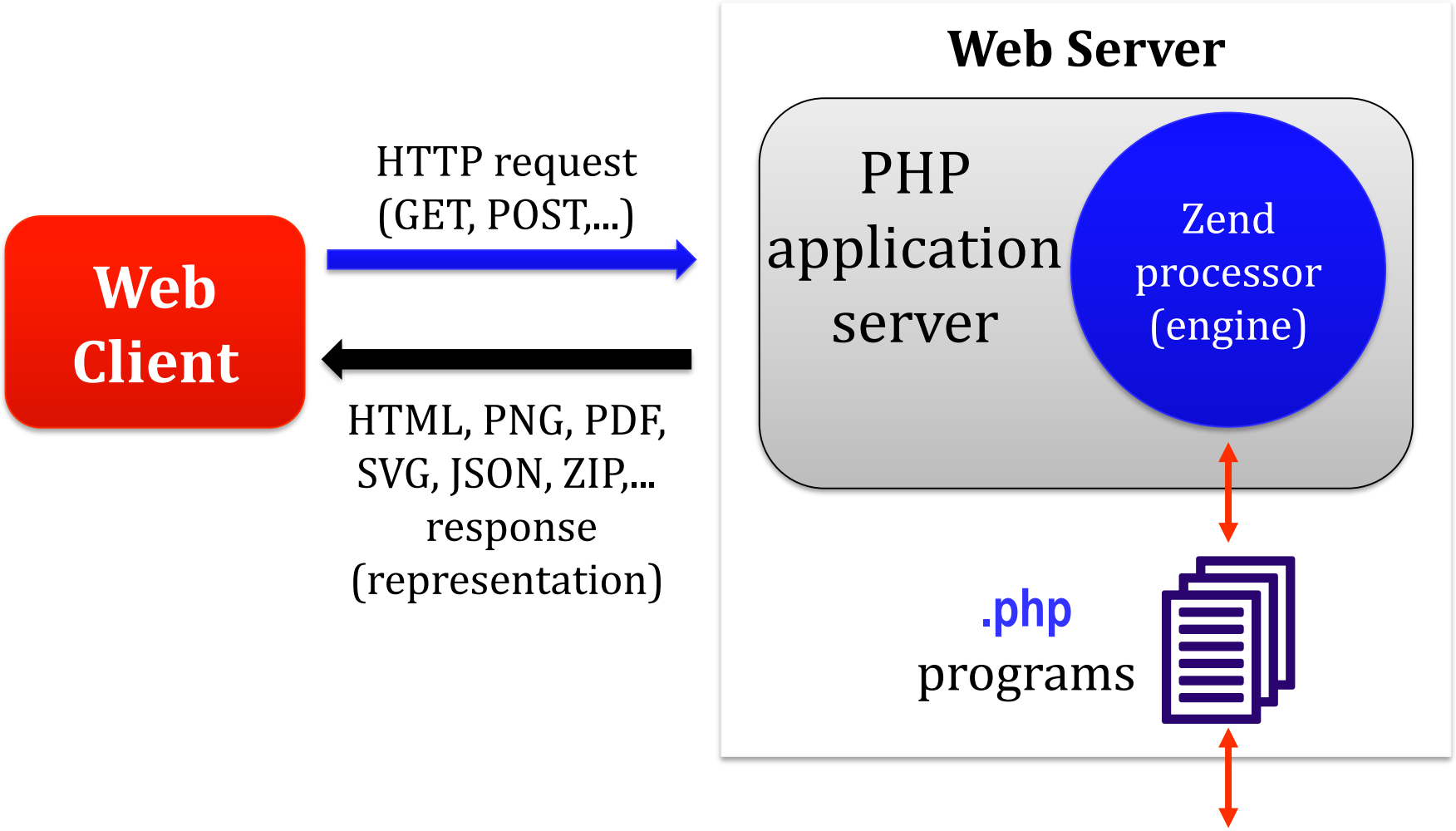
Freely available – open source – for various platforms
(FreeBSD, Linux, Windows, macOS etc.)
and Web servers: Apache, IIS, nginx,…

www.php.net
www.zend.com

PHP processor (engine) mode of operation



PHP processor (engine) mode of operation



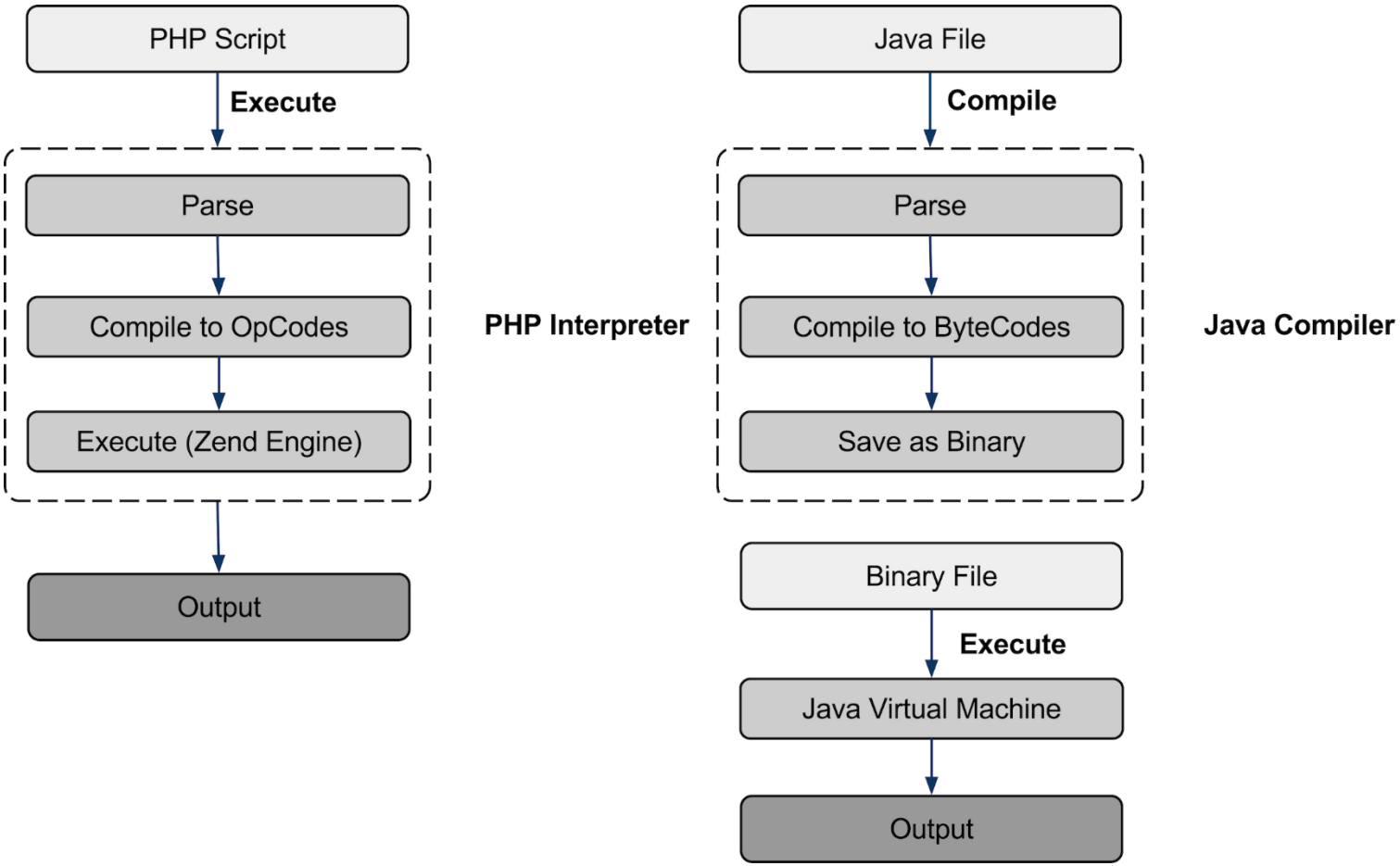
possibly, various extensions (modules) can be integrated

www.php.net/manual/en/extensions.php

php: characteristics

A PHP program is interpreted by Zend Engine 2 which generate internal instructions – opcodes

www.php.net/manual/en/internals2.opcodes.php
www.phpinternalsbook.com/



important phases of interpreting PHP programs *vs.* compiling Java code

PHP code is interpreted each time
when it must be executed

advanced

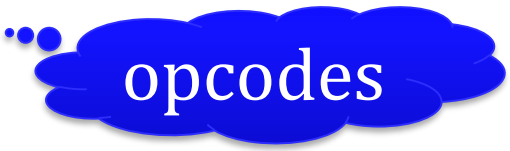
transforming PHP code into opcodes

tinyurl.com/zn6c53x

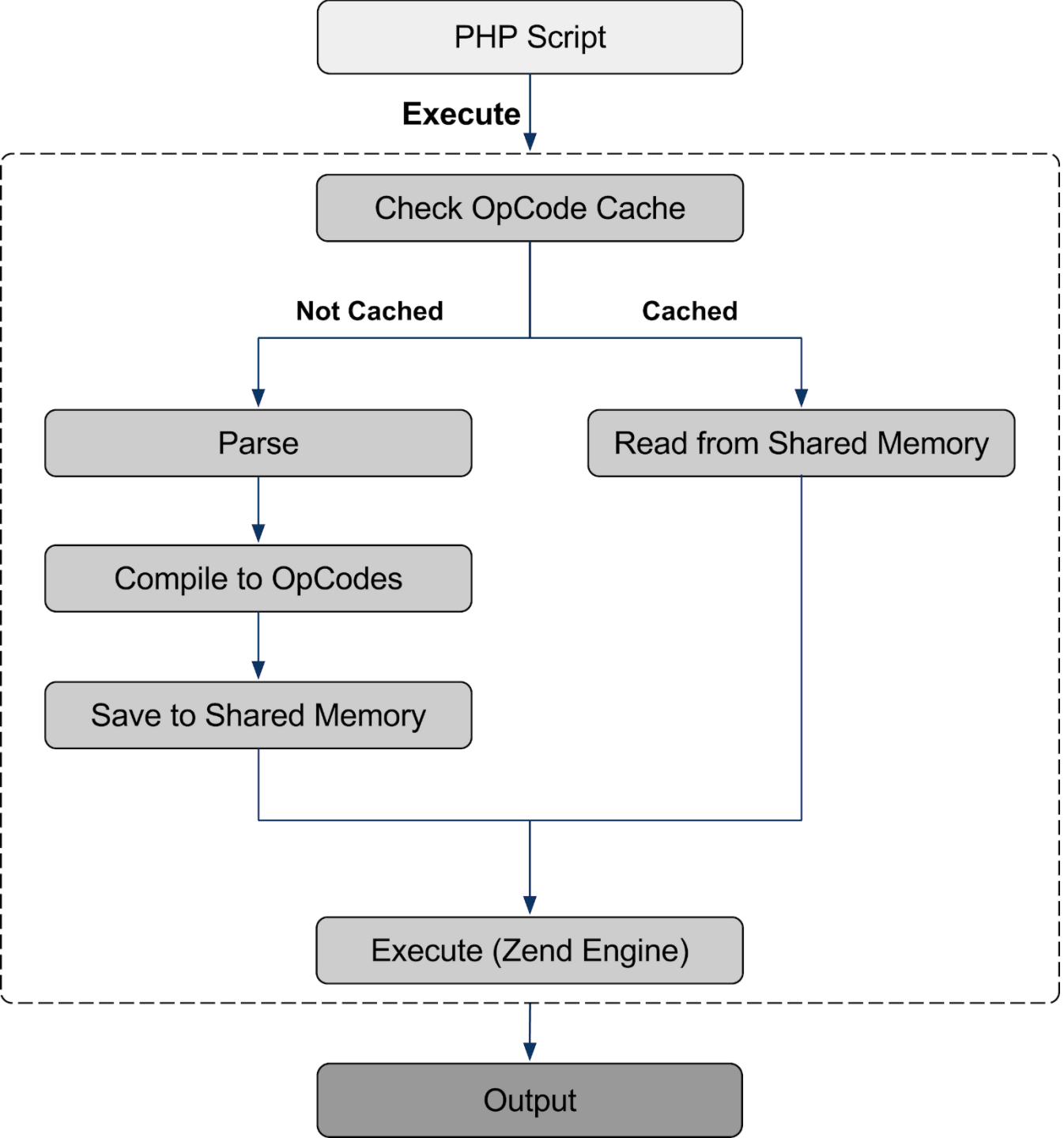
```php
<?php
class Greeting {
    public function sayHello ($to)
    {
        echo "Hello $to";
    }
}

$greeter = new Greeting ();
$greeter->sayHello ("World");
?>
```

```
line      # *    op                fetch          ext      return     operands
-------------------------------------------------------------------------------
   3      0  >   EXT_NOP
          1       RECV                                       !0
   5      2       EXT_STMT
          3       ADD_STRING                                 ~0      'Hello+'
          4       ADD_VAR                                    ~0      ~0, !0
          5       ECHO                                               ~0
   6      6       EXT_STMT
          7     > RETURN                                             null
```



advanced



for efficiency, the opcodes are stored into a shared memory

```
precision = 14            ; precision for float values – details at php.net/precision
safe_mode = Off           ; processing control – study php.net/safe-mode
disable_functions = "allow_url_fopen, eval" ; unallowed functions (e.g., security reasons)
max_execution_time = 30       ; max. number of seconds allocated to program execution
memory_limit = 128M  ; max. size of the memory allowed for a script
post_max_size = 8M    ; max. size of data transmitted via POST method
default_mimetype = "text/html" ; default MIME type for the content generated by a script
file_uploads = On         ; file uploads are permitted
upload_max_filesize = 32M    ; max. size of an uploaded file
session.use_cookies = 1       ; Web sessions will be stored in cookies
session.name = PHPSESSID ; default cookie name regarding Web sessions
...
; specifying the extensions loaded during the PHP server application initialization
extension_dir="./php/extensions" ; directory including extensions
extension=curl                ; HTTP + other Internet protocols using libcurl library
extension=pdo_sqlite          ; support for SQLite via PDO (PHP Data Objects)
extension=mysqli              ; support for MySQL
...
```

various behaviors of PHP platform,
including loaded extensions (.so/.dll shared libraries),
can be globally configured by using the **php.ini** file

php: characteristics

PHP program execution behavior can be adjusted via **declare** directive – possibly, for a block of code

www.php.net/manual/en/control-structures.declare.php

```
// charset encoding used for generated content
declare (encoding='UTF-8');

// strict datatype checking – PHP 7+
declare (strict_types=1);
```

advanced

php: characteristics

To increase performance, the just-in-time compilation (JIT) could be adopted

HHVM – HipHop Virtual Machine (Facebook)

PHP source-code ▸ opcodes ▸ machine code (*e.g.*, x86-64)

JIT

www.hhvm.com • github.com/facebook/hhvm/

advanced

php: characteristics

To increase performance, the just-in-time compilation (JIT) could be adopted

HHVM – HipHop Virtual Machine (Facebook)

includes a high-performance server: Proxygen
allows the upload of extensions written in PHP/C++
used by Baidu, Box, Etsy, Facebook, Wikipedia,…

advanced

php: characteristics

To increase performance, the just-in-time compilation (JIT) could be adopted

PHP 8 offers native support

www.zend.com/blog/exploring-php-8
www.zend.com/blog/exploring-new-php-jit-compiler

php: characteristics

User interaction:

retrieving the values entered into Web form fields
cookies
sessions
user authentication

access to global variables – created “on the fly”

php: characteristics

Features regarding Web technologies:

URL processing
HTTP support – including cURL
caching via memcached
Web services development by using SOAP and REST

…and others

php: characteristics

Support for database access:

abstract layers

DBAL (DataBase Abstraction layer)
iODBC (Independent Open DataBase Connectivity)
PDO (PHP Data Objects)

www.phptherightway.com/#databases_abstraction_layers

php: characteristics

Support for database access:

specific to a database server

relational: DB2, MySQL, Oracle, PostgreSQL, SQLite,…
NoSQL – *e.g.*, MongoDB

consult **www.phptherightway.com/#databases**

php: characteristics

Resource content processing:

audio files – via various libraries: ktaglib, oggvorbis, etc.
bzip2, LZF, RAR, ZIP, ZLIB archives
PDF documents
graphical content in various formats
JSON files
XML documents – creating, processing, validating, etc.
information regarding credit cards
...

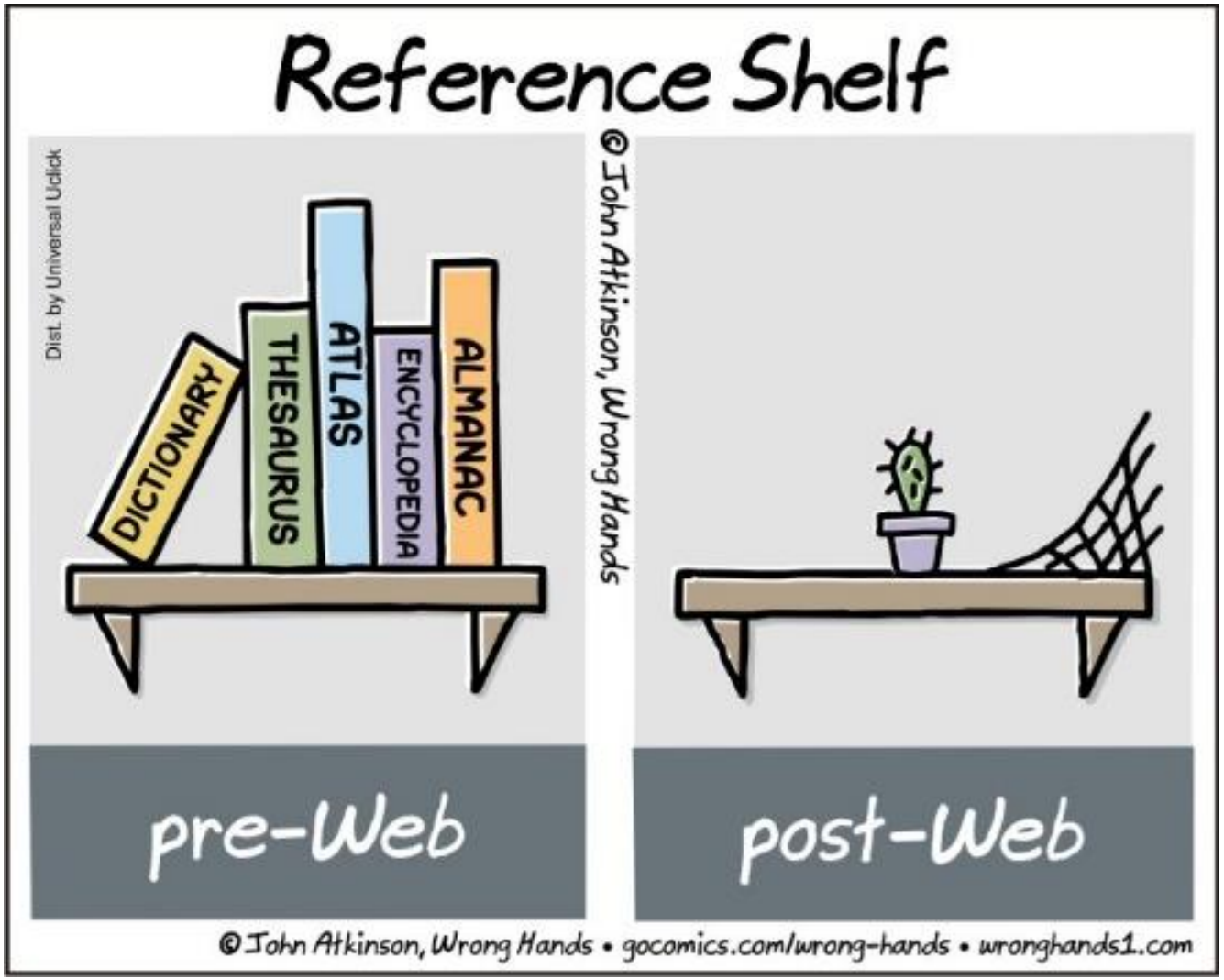
php: characteristics

Support for system resources + Internet:

file systems, including FTP
processes – with Libevent, PCNTL, pthreads,...
event processing – via Event
sockets
e-mail – *e.g.*, IMAP, POP3

...and many others

(instead of) break



wronghands1.files.wordpress.com/2020/01/reference-shelf.jpg

PHP as procedural programming language

php: characteristics

Procedural programming

paradigm based on procedure calls (routines, functions) containing a series of steps for making the calculations

procedural languages are imperative, using instructions (commands) that change the state of the program

examples: FORTRAN (1954), ALGOL (1958), BASIC (1964), Pascal (1970), C (1972), Ada (1978)

php: data types – scalars

boolean

TRUE or FALSE

php: data types – scalars

integer

values specified in…

10 (*decimal*)	3734
16 (*hex*)	0xE96
8 (*octal*)	07226
2 (*binary*)	0b111010010110

www.php.net/manual/en/language.types.integer.php

php: data types – scalars

float

real numbers
usually, conforming to IEEE 754 (double precision)

the size of the representation is platform dependent
(the precision can be adjusted in the config file php.ini)

www.php.net/manual/en/language.types.float.php
floating-point-gui.de

php: data types – scalars

float

special value:
NAN (not a number) constant

useful predefined functions:
is_nan()
is_finite()
is_infinite()

php: data types – scalars

string

ASCII character strings
(starting with PHP 7, native Unicode support exists)

php: data types – scalars

string

ASCII character strings
(starting with PHP 7, native Unicode support exists)

escape characters can be used, like
\n (New Line) **\r** (Carriage Return) **\t** (Tab)
\\ (Backslash) **\$** (Dollar) \" (Double Quote) \' (Quote)

php: data types – scalars

string

common delimiters:
" or '

string size cannot be more than 2 GB

php: data types – composites

array

association between values (of any type)
and keys (of integer or string types)

php: data types – composites

array

there is no clear distinction
between indexed and associative arrays

an array could represent various data structures:
list (vector), associative array - hash (implementation of an association of values - mapping), dictionary, collection, stack, queue,…

php: data types – composites

array

```
// an indexed array (vector of values) – initial syntax
$gifts = array (“watch", “cookie", “necklace", "axe");

// associative array – <key, value> pairs
array ( "name" => "Tux", "size" => 17, "offer" => TRUE );

// simplified syntax (starting with PHP 5.4)
[ "name" => "Tux", "size" => 17, "offer" => TRUE ];
```

php: data types – composites

object

instance of a class

created with the **new** operator

php: data types – special

resource

denotes a reference to an external resource

examples: bzip2, curl, ftp, gd, mysql link, mysql result, pdf document, printer, stream, socket, xml, zlib

a resource is created by using specific functions
e.g., a stream resource is initiated by the fopen() function and used by fread(), feof(), fgets() functions

php: data types – special

resource

denotes a reference to an external resource

predefined functions:
is_resource()
get_resource_type()

details at www.php.net/manual/en/resource.php

php: data types – special

null

specifies the NULL value
representing a variable that has no value

useful functions:
is_null()
unset()

php: variables

Variables have names composed by letters, digits, and _ characters prefixed by the **$** symbol

can store values – having a specific data type – or references (specified with **&**)

www.php.net/manual/en/language.variables.php

php: variables

Variables are created “on the fly”
data type is inferred according to the context

data type automatic conversion (type casting)
is similar to the C language

```
$age = 21;                      /* a variable of Integer type */
$connected = TRUE;              # a Boolean datatype one
$prefer["color"] = "gray";      // an associative array
```

```php
$nume = 'Tux';

// different behavior regarding the substitution of the actual
// value of a variable depending on the delimiters used
// for strings
echo "Salut $nume!\n";
echo 'Salut $nume!\n';
```

-> Salut Tux!
Salut $nume!\n

</> source code

```php
<?php
$nume = 'Tux';

echo "Salut $nume!\n";
echo 'Salut $nume!\n';
```

input Output

Success #stdin #stdout 0.02s 24312KB

```
Salut Tux!
Salut $nume!\n
```

online execution of PHP code via Ideone Web tool

php: variables

Useful predefined functions:

var_dump()

settype()

is_bool(), is_int(), is_float(), is_array(), is_string()

is_scalar(), is_numeric()

...and others

php: variables

Scope (variables visibility)

in order to be accessed in the entire program,
the variables should be declared as **global**

php.net/manual/en/language.variables.scope.php

```php
$score = 33;

function getScore () {
   echo "Current score: " . $score;
}

getScore();
```

- Undefined variable: score in prog.php on line 4

```php
$score = 33;

function getScore () {
   global $score;
   echo "Current score: " . $score;
   // similar to $GLOBALS["score"]
}

getScore();
```

- Current score: 33

php: variables

Scope (variables visibility)

a variable can also be declared as **static**

exists only in a local scope (*e.g.*, inside a function),
but it does not lose its value
when program execution leaves this scope

php: variables

Allocated memory is automatically freed (garbage collection)

each variable has a container associated with it in memory (*zval*)

see xdebug_debug_zval() function provided by Xdebug extension

www.php.net/manual/en/features.gc.php

php: predefined variables

Variables available in the whole program (superglobals)

$GLOBALS [ ]

an associative array containing references to all variables defined as global

php: predefined variables

$_SERVER [ ]

$_GET [ ] $_POST [ ] $_FILES [ ] $_REQUEST [ ]

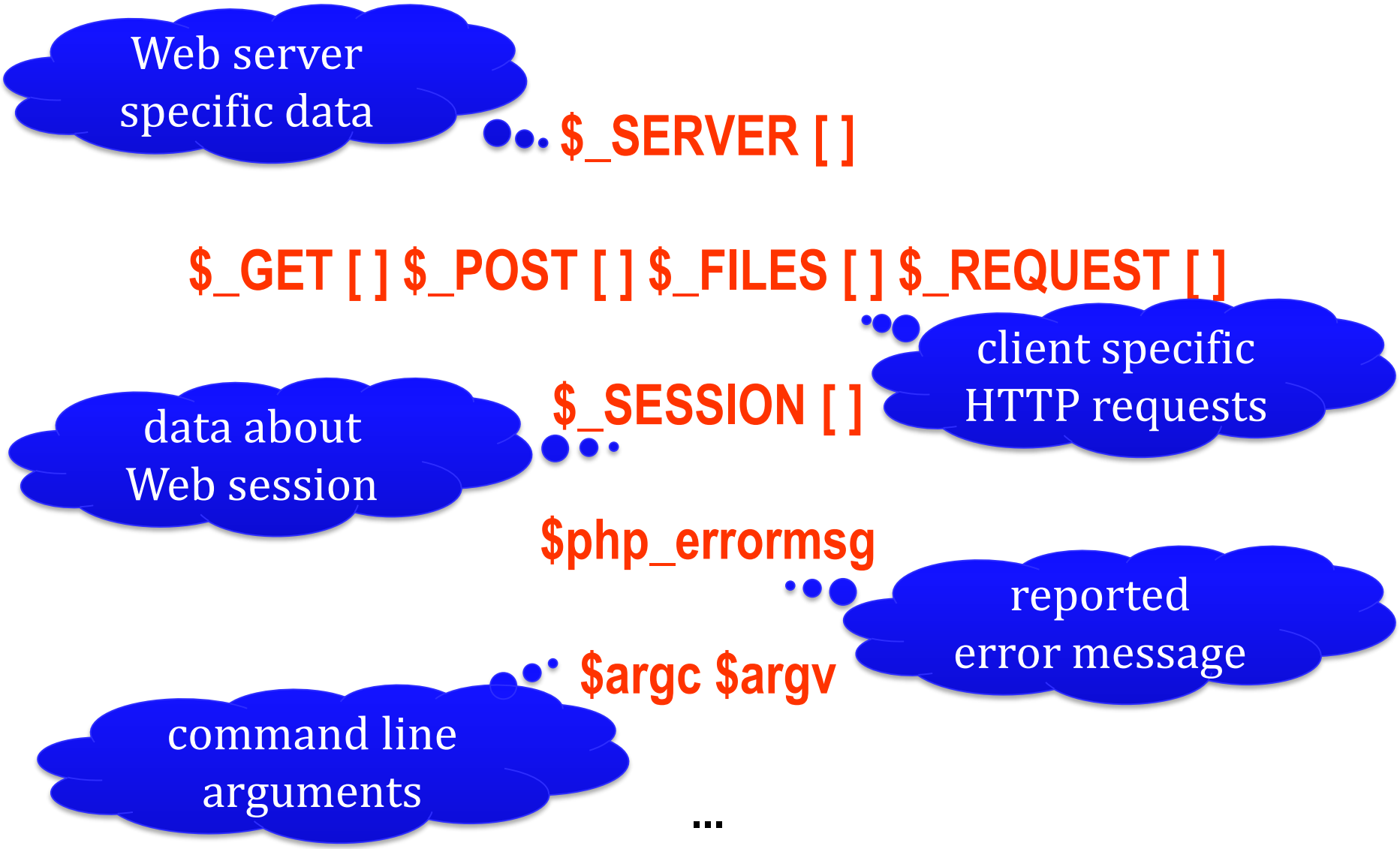
$_SESSION [ ]

$php_errormsg

$argc $argv

...

php: predefined variables



php: constants

Specified with **define ( )**

globally available in the program

define (string $name , mixed $value) : bool

```
define ("MIN_DIMENS", 13);
```

www.php.net/manual/en/function.define.php

php: predefined constants

Examples:

PHP_VERSION
PHP_OS
PHP_EOL
PHP_INT_MAX
PHP_INT_SIZE
DIRECTORY_SEPARATOR
true
false
null

php: predefined constants

Error reporting control:

E_ERROR	fatal errors (script execution is ended)
E_WARNING	warnings
E_PARSE	code processing errors (parsing)
E_NOTICE	notifications at runtime
E_STRICT	suggestions on code improvement
E_DEPRECATED	notifications about deprecated aspects

www.php.net/manual/en/errorfunc.constants.php
www.phptherightway.com/#errors_and_exceptions

advanced

php: predefined constants

Runtime environment offers access to “magical” constants
their values can be used in a given program

__LINE__
__FILE__
__DIR__
__FUNCTION__
__CLASS__
__TRAIT__
__METHOD__
__NAMESPACE__

php: operators

Majority of them are similar to the C language ones

arithmetic: **+ - * / % ++ --**
value assignation: **=** and **=>** (for arrays)
reference: **&**
reference assignation: **=&**
bit-wise: **& | ^ << >>**
comparisons: **== === != <> !== < > <= >= ?: ?? <=>**
error reporting control: **@**
logic (Booleans): **and or xor ! && ||**
string (concatenation) – same as Perl: **. .=**

php: operators

In PHP 7+, new operators can be used:

<=> (spaceship)
comparing expressions (of a scalar type),
and returning -1, 0 or 1

```
echo 15.5 <=> 15.5;      // 0  (equality)
echo 15.5 <=> 16.5;      // -1 (right value is greater)
echo 17.5 <=> 15.5;      // 1  (left value is greater)
```

php: operators

In PHP 7, new operators can be used:

?? (null coalescing)
offers the value of first operand if exists and it's not NULL,
otherwise returns the value of second operand

```
// as user name, we'll use the value provided by a Web form
// (retrieved with GET or POST);
// if it doesn't exist, the value is 'tux'
$username = $_GET['user'] ?? $_POST['user'] ?? 'tux';
```

php: control statements

if, switch, while, do, for, break, continue
similar to the C instructions

```
if (!$name) {
  echo ("No name was given…");
} else {
  echo ("Welcome, " . $name . "!\n");
}
```

php: example

```
<?php
  // filling up an array with values from 1 to 10
  for ($index = 1; $index <= 10; $index++) {
    $values[$index] = $index;
  }
  // computing the sum of values
  $sum = 0;
  foreach ($values as $item)
    $sum += $item;
  /* showing obtained sum on the standard output
     to be sent to the Web client */
  echo ("<p>Sum of first 10 numbers is <strong>" .
          $sum . "</strong>.</p>");
?>
```

Invoking (running) the PHP program
from the command line:

saving source-code into a text file – **values.php**

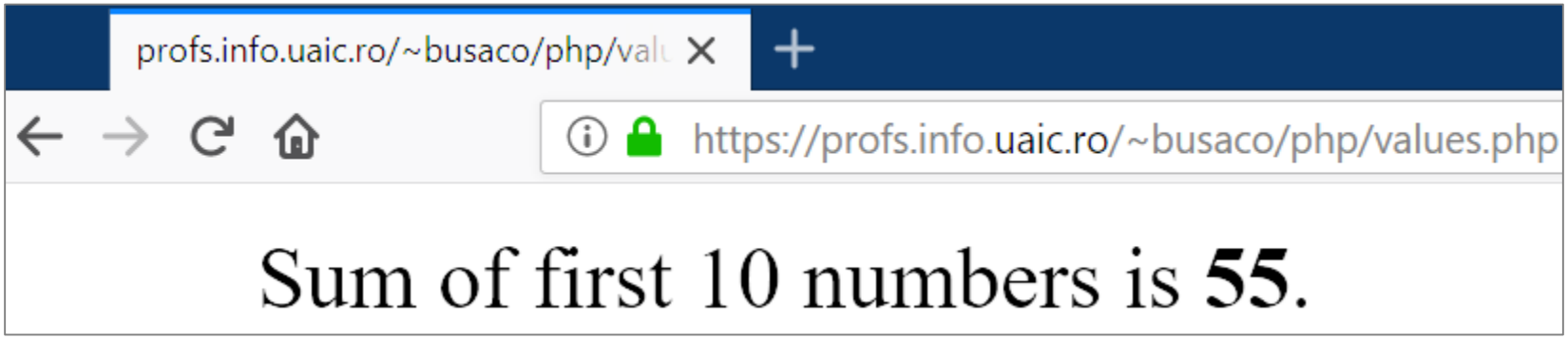
calling PHP interpreter from the command line interface

```
php values.php
<p>Sum of first 10 numbers is <strong>55</strong>.</p>
```

Invoking (running) the PHP program on the Web server:

placing the source code – having read rights

in the browser, specifying the URL to the program to be invoked via GET method of the HTTP protocol



result generated by the script

php: control statements

Including source-code from other files (support for modularization)

include

searches the source file in predefined directories specified via include_path and evaluates it

if the file does not exist, a warning is generated

include_once – to include it only once

php: control statements

Including source-code from other files (support for modularization)

require

searches the source file in predefined directories specified via include_path and evaluates it

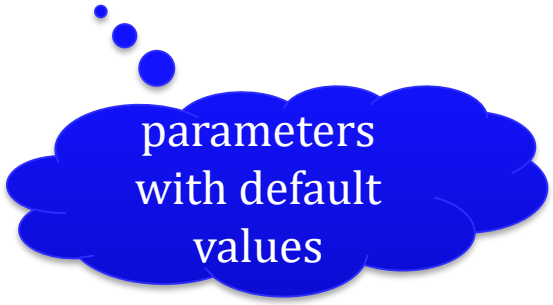
if the file does not exist, a fatal error is generated

require_once – to include it only once

php: functions

User defined functions:

```
function sendMessage ($to="", $from="", $subject="Web") {
   // body…
}
```



php.net/manual/en/language.functions.php

```php
define ('MAX', 10);                    // maximum number of values

function square ($number) {  // a function to square a number
  return $number * $number;
}

$number = 0;
while ($number < MAX) {
  $number++;                           // incrementing…

  if ($number % 2)                     // is odd number...
    continue;                          // continuing to next iteration
  // it is an even number, so showing its square
  echo "Square $number is " . square ($numar) . "\n";
} // end of while
```

php: functions

User defined functions:

a function name is case-insensitive

parameters can be specified by reference – prefixed by &

for PHP 5.6+, a variable number of parameters
is indicated by ...

php.net/manual/en/functions.arguments.php

The functions are visible anywhere in the program:

```
infomagic ();

function infomagic () {
  printf("<p>I'm <code>%s()</code> and I'm executing line %d
    from <code>%s</code>.</p>", __FUNCTION__, __LINE__, __FILE__);
  printf("<Using PHP <code>%s</code> on <code>%s</code>
   with the Web server <code>%s</code>.</p>",
   PHP_VERSION, php_uname(), $_SERVER['SERVER_SOFTWARE']);
}
```

The functions are visible anywhere in the program:

```
infomagic ();

function infomagic () {
  printf("<p>I'm <code>%s()</code> and I'm executing line %d
    from <code>%s</code>.</p>", __FUNCTION__, __LINE__, __FILE__);
  printf("<Using PHP <code>%s</code> on <code>%s</code>
   with the Web server <code>%s</code>.</p>",
   PHP_VERSION, php_uname(), $_SERVER['SERVER_SOFTWARE']);
}
```

- I'm infomagic() and I'm executing line 15 from /opt/lampp/htdocs/magic.php.
  Using PHP 7.4.2 on Linux debian 4.9.0-11-amd64 #1 SMP
  Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64 with the Web server
  Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.2
  mod_perl/2.0.8-dev Perl/v5.16.3.

```php
<?php
declare (strict_types=1);

// arguments should be integers, returned value must be of integer type
function add (int ...$numbers): int {
    $sum = 0;
    foreach ($numbers as $number) {
        $sum += $number;
    }
    return $sum;
}

echo add (7, 3, 74, 1);
echo add (pi (), '?');
?>
```

```
85
Fatal error: Uncaught TypeError:
Argument 1 passed to add() must
be of the type integer, float given
Next TypeError: Argument 2 passed
to add() must be of the type integer,
string given
```

PHP 7+: data type can be specified for each argument and for the value returned by the function (scalar type declarations)

php: functions

Beginning with PHP 5.3,
anonymous functions can be defined

- functional programming (*e.g.*, closures)

```
$hello = function ($name) {        // variable type is function
    printf ("Hello %s...\n", $name);
};

$hello ('world');
$hello ('Tuxy');
```

see www.phptherightway.com/pages/Functional-Programming.html

php: functions

Anonymous function can be specified shorter
Starting with PHP 7.4 (launched in November 2019)

- ***short closures***

concept similar to arrow functions

functional construction available
in JavaScript language (ECMAScript 6 – ES6)

```php
declare (strict_types=1);                // strict data type check
header ('Content-type: text/plain'); // send unformatted text to the client

// short declaration of an anonymous function
// which has one string argument and returns an int
// variable of function type: fn(argument) => body
// (for the body, a single expression is allowed; 'return' is not allowed)
$hello = fn(string $name): int => printf("Hello %s! [la %s].\n",
                                         $nume, date("j-m-Y H:i:s T", time()));
$hello('world');
sleep (2); // wait 2 seconds...
$hello('Tuxy');
```

```
Hello world! [at 14-03-2022 01:37:59 PST].
Hello Tuxy! [at 14-03-2022 01:38:01 PST].
```

php: predefined functions

mathematical & conversion
character string manipulation
array processing
access to resources and file processing
database manipulation
regarding network connections
cryptographic
XML, PDF, JPEG,... resource processing
operating system specific
general purpose

details at **php.net/manual/en/funcref.php**

php: predefined functions

Math:

abs(), mod(), fmod()
ceil(), floor(), round(), max(), min()
exp(), log10(), log()
pow(), sqrt()
sin(), cos(), tan(), asin(), …, sinh(), …, pi()
rand(), srand()
bindec(), octdec(), dechex(),…, base_convert()
is_finite(), is_infinite(), is_nan()

php.net/manual/en/refs.math.php

php: predefined functions

Strings:

echo(), print(), printf(), sprintf() etc.
strlen(), chr(), ord(), substr(), strstr(), strpos(),...
strcmp(), strcasecmp(), strnatcmp() etc.
strcat(), str_replace(), str_ireplace(), strrev() etc.
trim(), ltrim(), rtrim()
explode(), implode(), split(), join(), strtok()

PHP 8: str_contains(), str_starts_with(), str_ends_with()

details about text processing:
php.net/manual/en/refs.basic.text.php

php: predefined functions

Regular expressions:

conforming to POSIX standard
ereg(), ereg_replace(), split() etc.

Perl compatible – PCRE: www.pcre.org
preg_filter(), preg_grep(), preg_match(), preg_split(),…

PHP 8+ (2020): new match() construction
www.php.net/manual/en/control-structures.match.php

php: predefined functions

Arrays:

array_count_values(), array_search(), array_filter(),
array_slice(), array_chunk()
array_fill(), array_combine(), array_shift(),
array_reverse(), array_multisort(), array_sum(),…
array_merge(), array_intersect(), array_diff()
array_keys(), array_key_exists()
array_push(), array_pop()
array_map(), array_reduce()

php.net/manual/en/book.array.php

advanced

```php
/* filters certain values from an array
   by using a programmer defined function */
function value_less_than ($number) {
   // returns an expression of function type
   return function ($item) use ($number) { // a closure (functional approach)
      return $item < $number;
   };
}

$marks =  [ 7, 8, 9, 10, 7.5, 3, 10, 8.75, 4 ];
// using array_filter(), a predefined function, on marks array
// to obtain the values less than a given value (here: 7)
$values = array_filter ($marks, value_less_than (7));

print_r ($values);
```

```
Array
(
    [5] => 3
    [8] => 4
)
```

also, study wiki.php.net/rfc/closures

advanced

In PHP 7.4+ the operator ... can be used within arrays (similar with ECMAScript 2018)

```php
$web = [ 'JS', 'PHP' ];
$whims = [ 'Lua', 'Rust', 'Scala' ];
$languages = [ 'C#', ...$web, 'Java' ];

print_r ($languages);
// fusion – similar with array_merge()
print_r ([ ...$whims, ...$web ]);
```

```
Array
(
    [0] => C#
    [1] => JS
    [2] => PHP
    [3] => Java
)
Array
(
    [0] => Lua
    [1] => Rust
    [2] => Scala
    [3] => JS
    [4] => PHP
)
```

php: predefined functions

Character manipulation:

ctype_digit(), ctype_xdigit(), ctype_print(),
ctype_punct(), ctype_space(),…
ctype_alpha(), ctype_alnum(), ctype_lower(),
ctype_upper()

php: predefined functions

Date & time:

getdate(), localtime(), gettimeofday(), time(), etc.
date(), idate(), gmdate(),…
checkdate()
strftime(), strtotime()

see also Calendar, DateTime, HRTime extensions

php: predefined functions

PHP variables:

empty(), isset(), unset()
strval(), print_r(), var_dump()
serialize(), unserialize()

also, consult php.net/manual/en/book.var.php

php: predefined functions

Files and directories:

using FILE data type – similar to C language:
fopen(), fread(), fscanf(), fgets(), fwrite(), fprintf(),
fseek(), ftell(), feof(), fclose(), ftruncate(), fstat(),...

file(), copy(), rename(), delete(),
move_uploaded_file(), tmpfile()

file_exists(), filesize(), filetype(), fileperms(),…, stat()

is_dir(), is_file(), is_readable(), is_writeable(),…

chdir(), mkdir(), rmdir()

disk_free_space(), disk_total_space()

study **php.net/manual/en/refs.fileprocess.file.php**

php: predefined functions

URLs:

urldecode(), urlencode(), parse_url()
base64_decode(), base64_encode()
http_build_query()

processing the components of a URL:

```
define('URL', 'http://somewhere.info:8080/offer/toys/product/?name=Tux&size=17#offer');

header('Content-type: text/plain'); // content sent to the client will be plain text

// dividing the URL into components of an associative array
$WebAddress = parse_url(URL);

foreach(array_keys($WebAddress) as $component) {
  printf("%s=%s\n", $component, $WebAddress[$component]);
}

// processing the URL's query string
// each parameter will be stored in $parameters array
parse_str($WebAddress['query'], $parameters);

if ($parameters['size'] < 13) {
  echo "Toy is not ok...\n";
} else {
  echo "Toy is ok (" . $parameters['size'] . ")
          and has the name " . $parameters['name'] . ".\n";
}
```

processing the components of a URL:

```
define('URL', 'http://somewhere.info:8080/offer/toys/product/?name=Tux&size=17#offer');

header('Content-type: text/plain'); // content sent to the client will be plain text

// dividing the URL into components of an associative array
$WebAddress = parse_url(URL);

foreach(array_keys($WebAddress) as $component) {
  printf("%s=%s\n", $component, $WebAddress[$component]);
}

// processing the URL's query string
// each parameter will be stored in $parameters array
parse_str($WebAddress['query'], $parameters);

if ($parameters['size'] < 13) {
  echo "Toy is not ok...\n";
} else {
  echo "Toy is ok (" . $parameters['marime'] . ")
          and has the name " . $parameters['nume'] . ".\n";
}
```

```
scheme=http
host=somewhere.info
port=8080
path=/offer/toys/product/
query=name=Tux&size=17
fragment=offer
Toy is ok (17)
and has the name Tux.
```

php: predefined functions

Web resource processing (HTML, JSON):

nl2br(), htmlentities(), htmlspecialchars(), strip_tags()

get_browser(), show_source(), highlight_string(),...

json_encode(), json_decode(), json_last_error()

(de/en)coding internal data ↔ JSON data:

```php
// songs from the album 'Animals' by Pink Floyd (1977) expressed in Unicode
$animals = [ "🐷 on the Wing #1", "Dogs 🐕",
  [ "🐖🐖🐖" => "Pigs (Three Different Ones)" ],
  "Sheep 🐑", "🐷 on the Wing #2" ];

$json = json_encode($animals);   // encoding (serializing) in JSON

$animals2 = json_decode($json);  // decoding JSON data

// information regarding used variables
var_dump($animals, $animals2, $json);
```

$animals

```
array(5) {
  [0]=>
    string(19) "🐷 on the Wing #1"
  [1]=>
    string(16) "Dogs 🐕‍🦺"
  [2]=>
    array(1) {
    ["🐖🐖🐖"]=>
      string(27) "Pigs (Three Different Ones)"
  }
  [3]=>
    string(10) "Sheep 🐑"
  [4]=>
    string(19) "🐷 on the Wing #2"
}
```

$animals2

```
array(5) {
  [0]=>
    string(19) "🐷 on the Wing #1"
  [1]=>
    string(16) "Dogs 🐕‍🦺"
  [2]=>
    object(stdClass)#1 (1) {
    ["🐖🐖🐖"]=>
      string(27) "Pigs (Three Different Ones)"
  }
  [3]=>
    string(10) "Sheep 🐑"
  [4]=>
    string(19) "🐷 on the Wing #2"
}
```

$json

string(191) "["\ud83d\udc37 on the Wing #1","Dogs \ud83d\udc15\u200d\ud83e\uddba",
{"\ud83d\udc16\ud83d\udc16\ud83d\udc16":"Pigs (Three Different Ones)"},
"Sheep \ud83d\udc11","\ud83d\udc37 on the Wing #2"]"

php: predefined functions

Support for cryptographic operations:

password hashing – password_*() functions

useful extensions

- CSPRNG (generating pseudo-random numbers – in PHP 7+)
- Hash (hash_*() digest functions)
- OpenSSL (functionalities for SSL/TLS)
- Sodium (advanced encrypt/decrypt operations – in PHP 7.2+)

details in **php.net/manual/en/refs.crypto.php**

php: predefined functions

Support for graphical content (raster/vector):

preinstalled extensions

Cairo – vector/raster processing: www.cairographics.org

EXIF – access to JPEG meta-data

GD – raster processing (GIF, JPEG, PNG): libgd.github.io

ImageMagick – multi-format processing: www.imagemagick.org

php.net/manual/en/refs.utilspec.image.php

php: predefined functions

Other useful functions:

die(), eval(), exit(), sleep(), usleep(), time_sleep_until()

uniqid(), sys_getloadavg()

php_info(), php_check_syntax()

advanced

php: other features

SPL (Standard PHP Library)
access to standard ways of data processing

defined data structures:
SplStack, SplQueue, SplHeap, SplPriorityQueue,...

iterators:
ArrayIterator, FilesystemIterator, RegexIterator etc.

www.php.net/spl
www.phptherightway.com/#standard_php_library

advanced

php: other features

Internationalisation & localization of applications

lexical checking (spelling): Enchant

support for translating messages: Gettext

internationalisation (i18n): intl
classes: NumberFormatter MessageFormatter IntlDateFormatter IntlTimeZone IntlCalendar Locale

www.php.net/manual/en/refs.international.php

advanced

php: other features

Other preinstalled basic extensions

In PHP 7+, efficient data structures can be used

interfaces:
Collection Hashable Sequence

classes:
Vector Deque Map Pair Set Stack Queue PriorityQueue

www.php.net/manual/en/book.ds.php

advanced

php: other features

Other preinstalled basic extensions

YAML – support for *YAML Ain't Markup Language*
Swoole – network access: TCP, UDP, HTTP, WebSocket
Streams – work with data streams
SeasLog – journaling features
GeoIP – geographical localisation
Parsekit –PHP opcodes analysis

www.php.net/manual/en/refs.basic.other.php

advanced

php: other features

Process execution control

extensions of interest
(some available only on Linux systems)

Eio, Ev, Expect, Libevent, PCNTL, POSIX, parallel, pthreads, pht, Semaphore, Shared Memory, Sync

www.php.net/manual/en/refs.fileprocess.process.php

advanced

php: other features

Process execution control

Asynchronous code execution cand be done via additional libraries

example:

ReactPHP

facilitates asynchronous – non-blocking – event-based input/output operations

reactphp.org

advanced

php: other features

Program execution from command line

PHP CLI

php.net/manual/en/features.commandline.php

advanced

php: other features

PHP as a Web server module (Apache, NGINX)

e.g., by using **proxy_fcgi** and **php-fpm** (FastCGI Process Manager) modules – **php-fpm.org**

wiki.apache.org/httpd/php
www.nginx.com/resources/wiki/start/topics/examples/phpfcgi/

advanced

php: other features

PHP 5.4+ includes its own Web server

run with:

php –S localhost:8000 –t *phpwebapp/*

php.net/features.commandline.webserver

advanced

php: other features

Integration with other technologies/platforms

examples of extensions:
V8js – process and execute JavaScript code
via V8 processor (Google) used by Chrome and Node.js
www.php.net/manual/en/book.v8js.php

Lua + LuaSandbox – interpreting Lua programs
www.php.net/manual/en/book.luasandbox.php

How about PHP support
for object-oriented programming?

php: characteristics

Object-oriented programming

paradigm based on the concept of **object** – including **data** (attributes, properties) and **code** (methods, procedures)

usually objects interact with each other and represent **class** instances

examples: Smalltalk (1972), Objective-C (1984), C++ (1985), Python (1990), Java (1995), C# (2000)

php: classes

Class definition via **class**
and instantiation by using **new** operator

objects are treated similar to references
(a variable of object type contains a reference to an object, not a copy of it)

php.net/manual/en/oop5.intro.php
details in **php.net/language.oop5**

object-oriented programming – encapsulation

```
class Student { // specifying a class
  // properties (data members)
  private $year;
  private $email;
  public  $name;
  // public methods
  public function setYear ($aYear) {
     $this->year = $aYear;
  }
  public function getYear () {
     return $this->year;
  }
}
```

$this is a pseudo-variable
specifying a reference to the current object

object-oriented programming – encapsulation

```
class Student { // specifying a class
  // properties (data members)
  private $year;
  private $email;
  public  $name;
  // public methods
  public function setYear ($aYear) {
     $this->year = $aYear;
  }
  public function getYear () {
     return $this->year;
  }
}
```

```
// an object instance
$stud = new Student ();
$stud->setYear (2);
$stud->name = 'Tux';
print_r ($stud);
```

```
▸ Student Object
(
    [year:Student:private] => 2
    [name] => Tux
    [email:Student:private] =>
)
```

php: classes

Similar to C++, members – properties or methods – can be declared as

public
private
protected

object-oriented programming – inheritance

```
class CleverStudent extends Student {
    private $marks;    // obtained marks (property)

    public function setMarks ($m) {
        $this->marks = (array) $m;
    }
    public function getMarks () {
        return (array) $this->marks;
    }
}

$otherStud = new CleverStudent ();
// calling a method from base class
$otherStud->setYear (2);
// calling a method from derived class
$otherStud->setMarks (
   ['Web' => 10, 'SoftEng' => 9]
);
```

object-oriented programming – inheritance

```
class CleverStudent extends Student {
    private $marks;    // obtained marks (property)

    public function setMarks ($m) {
        $this->marks = (array) $m;
    }
    public function getMarks () {
        return (array) $this->marks;
    }
}

$otherStud = new CleverStudent ();
// calling a method from base class
$otherStud->setYear (2);
// calling a method from derived class
$otherStud->setMarks (
   ['Web' => 10, 'SoftEng' => 9]
);
```

```
print_r ($altStud);

 ▸ CleverStudent Object
(
    [marks:CleverStudent:private]
        => Array
        (
            [Web] => 10
            [SoftEng] => 9
        )

    [year:Student:private] => 2
    [name] =>
    [email:Student:private] =>
)
```

php: classes

Special methods:

constructors are named **__construct()**

destructors are named **__destruct()**

php: classes

Accessing static, constant or overloaded properties/methods

::

scope resolution operator (Paamayim Nekudotayim)

www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php

php: classes

Accessing static, constant or overloaded properties/methods

::

self – current class
parent – parent class

starting with PHP 7.4 (2019) the data type of the properties defined by a class can be explicitly specified

```php
declare (strict_types=1);

class Student {
  // properties having a type
  private int $year;
  private string $email;
  public  string $name;
  // etc.
}

$stud = new Student ();
$stud->name = true;
```

Fatal error: Uncaught TypeError: Cannot assign bool to property Student::$name of type string

Web tool for testing: **sandbox.onlinephpfunctions.com**

php: classes

Properties or methods declared as **static**
can be accessed without class instantiation

for examples, visit
www.php.net/manual/en/language.oop5.static.php

php: classes

Abstract classes/methods are declared with **abstract**

abstract classes can not be instantiated

each class having at least one abstract method
is considered abstract

abstract methods must be implemented in a child class
(specified by **extends**) of an abstract class

php: interfaces

Specifying methods that will be implemented by a class (similar to Java)

```
// interface regarding a content template
interface iTemplate {
    // setting a variable to be substituted with its value
    public function setVar ($name, $var);
    // getting the template representation
    public function getRep ($template);
}
```

details at php.net/manual/en/language.oop5.interfaces.php

```php
// a class implementing the interface
class Template implements iTemplate {
    // associative array including variables to be replaced by their values
    private $variables = array ();

    public function setVar ($name, $var) {
        $this->variables[$name] = $var;
    }

    public function getRep ($template) {
        foreach ($this->variables as $name => $val) {
            // replacing a variable name with its value
            $template = str_replace ('{' . $name . '}', $val, $template);
        }
        return $template;
    }
}
```

advanced aspects: www.phptherightway.com/#templating

php: predefined interfaces & classes

Traversable
Iterator
IteratorAggregate
Throwable
ArrayAccess
Serializable
Closure
Generator

www.php.net/manual/en/reserved.interfaces.php

php: predefined interfaces & classes

```
// example: Iterator interface
Iterator extends Traversable {

  // methods to be written by programmer
  // in the class implementing the interface
  abstract public mixed current ( void )
  abstract public scalar key ( void )
  abstract public void    next ( void )
  abstract public void    rewind ( void )
  abstract public boolean valid ( void )
}
```

php: reflection

Access to information regarding a class:

a PHP program can get data about classes, interfaces, functions, methods, extensions – *reverse engineering*

ReflectionClass implements Reflector

php.net/manual/en/book.reflection.php

php: reflection

```
$a_class = new ReflectionClass (‘CleverStudent');
// display information about the specified class
printf ("<p>Class <em>%s</em> extends %s
        and is declared in file <tt>%s</tt>.</p>",
        $a_class->getName (),
        var_export ($a_class->getParentClass (), 1),
        $a_class->getFileName ());
```

Class *CleverStudent* extends
ReflectionClass::__set_state(array('name' => '**Student**',))
and is declared in file /home/profs/busaco/html/php/introspect.php

php: traits

Trait

concept borrowed from Self language

available for PHP 5.4+

collection of methods that can be reused in other classes

www.php.net/manual/en/language.oop5.traits.php

php: traits

Trait

considered as a (C++) class template

in contrast to interfaces, offers implementations of methods, not just their signature

thus, support is provided for multiple pseudo-inheritance

aspecte formale: **scg.unibe.ch/research/traits**

```php
// traits (behaviors) that will be associated to 2D geometrical
    shapes
trait Rotating {
  public function rotate ($angle) {          // implements rotation
  }
}

trait Moving {
  public function moveTo ($x, $y) {          // move to other
    coordinates
  }
}

trait Filling {
  public function fill ($color) {            // performs filling
  }
}
```

```
abstract class Shape {  // geometrical shapes class
  public function draw() {
    echo ('I am drawing a ' . get_class());
  }
}

class Rectangle extends Shape {      // uses desired traits
  use Filling, Moving, Rotating;     // can be filled, moved,
    rotated

  public function transform () {     // plus, a specific
    transformation
  }
}
```

```
// Circle class can not be extended
final class Circle extends Shape {
  // a circle can be moved or filled
  use Moving, Filling;

  // constant declaration
  const PI = 3.1415265;

  // method specification
  public function computeArea () {
  }
}
```

php: traits

```
// 2 shapes (a circle and a rectangle) are instantiated
$aCircle = new Circle ();
$aRect = new Rectangle ();
$aCircle->draw ();
$aCircle->rotate ();                    // an error will be emitted
$aRect->draw ();
```

- I am drawing a Circle

  PHP Fatal error: Call to undefined method Circle::rotate()
  in /home/dMdWgn/prog.php on line 47

php: special properties

A class has special ("magical") properties associated that can be overloaded

__construct ()
__destruct ()

__toString ()

__get ()
__set ()

others at www.php.net/manual/en/language.oop5.magic.php

php: objects

Objects can be “cloned” via **clone**

Objects can be compared by using the **===** operator

php: objects

Functions for class & object manipulation

get_class() will return an object name,
instance of a given class

get_parent_class() gets the parent class of an object

method_exists() tests if a method exists
for a certain object

class_exists() tests the existence of a class

is_subclass_of() determines if a heritage relation
between two classes exists

php: exceptions

try catch throw

Exception class

similar to Java exceptions

details at www.php.net/manual/en/language.exceptions.php

php: namespaces

Used to avoid name collisions and to create aliases

declared with **namespace** (first line of a program)

php: namespaces

Used to avoid name collisions and to create aliases

declared with **namespace** (first line of a program)

example: **namespace Facebook;** // Facebook SDK for PHP

also study **www.phptherightway.com/#namespaces**

php: namespaces

Used to avoid name collisions and to create aliases

same namespace can be defined in multiple files

namespace hierarchies are permitted

```
namespace Project\Module\Submodule;

class SVGGen { ... };
```

refer with
Project\Module\Submodule\SVGGen

php: namespaces

Used to avoid name collisions and to create aliases

using a specific construct: **use**
(optionally, specifying an alias)

```
use Project\Module\Submodule;
```

php: namespaces

Used to avoid name collisions and to create aliases

using a specific construct: **use**
(optionally, specifying an alias)

```
use Project\Module\Submodule;
```

real examples:

```
use Facebook\Authentication\AccessToken;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
```

php: namespaces

Used to avoid name collisions and to create aliases

without any namespace definition, all class and function definitions are placed into the global space

```
namespace WebProject;
function cosh () {        // specifying our own function
   ...
   $value = \cosh (...); // calling a predefined function (from global space)
}
```

How about the features regarding Web interaction?

php: web interaction

Data transmitted by client (browser) are stored into (global) predefined associative arrays:

$_GET[ ] – data transmitted via GET

$_POST[ ] – data transmitted via POST

$_COOKIE[ ] – received cookies

$_REQUEST[ ] – data retrieved from client (content of **$_GET**, **$_POST** and **$_COOKIE**)

$_SESSION[ ] – session data

php: web interaction

Other useful global variables:

$_SERVER[ ]
offers info about Web server

$_SERVER['PHP_SELF'] indicates the PHP script name
$_SERVER['REQUEST_METHOD'] – used HTTP method
$_SERVER['HTTP_REFERER'] – URL referring the resource
$_SERVER['HTTP_USER_AGENT'] – client details

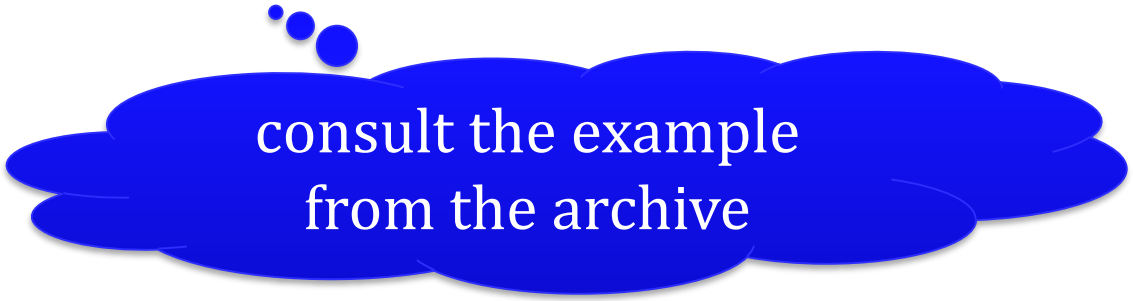
www.php.net/manual/en/reserved.variables.server.php

php: web interaction

Other useful global variables:

$_ENV[ ] – data provided by the runtime environment

$_FILES[ ] – data about the uploaded files
www.php.net/manual/en/features.file-upload.php



```
<!-- a Web form written in HTML -->
<form action="display.php" method="post">
        <input type="text" name="name" />
        <input type="text" name="age" />
        <input type="submit" value="Send" />
</form>
```

```
<!-- a Web form written in HTML -->
<form action="display.php" method="post">
      <input type="text" name="name" />
      <input type="text" name="age" />
      <input type="submit" value="Send" />
</form>
```

```
<?php
    // display.php – a program invoked via POST
    if (!$_REQUEST["name"])
      display ("Name is not specified!", "err");
    else
      display ("Name is" . $_REQUEST["name"]);
?>
```

each name field in the form represents a key in **$_REQUEST [ ]** associative array (depending on the HTTP method, it could be consulted by using **$_GET** or **$_POST**)

php: cookies – creation

Setting a cookie via **setcookie ()** function

setcookie (string $*cookie_name*
[, string $*value* = "" [, int $*expiration_time* = 0
[, string $*path* = "" [, string $*domain* = ""
[, bool $*secure_transf* = false [, bool $*just_http* = false]]]]]) : bool

php: cookies – creation

Setting a cookie via **setcookie ()** function

setcookie (string $*cookie_name*
[, string $*value* = "" [, int $*expiration_time* = 0
[, string $*path* = "" [, string $*domain* = ""
[, bool $*secure_transf* = false [, bool $*just_http* = false]]]]]) : bool

```
// persistent – set that the cookie will expire in 10 days
setcookie (‘color', 'tan', time() + 60 * 60 * 24 * 10);
// non-persistent – why?
setcookie ('authenticated', 'true');
```

php: cookies – expiration

Deleting a cookie:
cancel value and time;
eventually, also other cookie attributed

```
setcookie ($cookie_name, "", 0, "/", "");
```

detalii la www.php.net/manual/en/function.setcookie.php

php: cookies – access

A cookie is specified (accessed) like a variable of type associative array – $_COOKIE ['cookie_name']

```
<!-- the background color is given by the previously set
     cookie value -->
<style>
body {
  background: <?php echo $_COOKIE['color']; ?>
}
</style>
```

see the archive with PHP examples

php: web sessions

Session management: session_start(), session_register(), session_id(), session_unset(), session_destroy() functions

```
session_start (); // start a Web session
if (!isset ($_SESSION['visits'])) {
        $_SESSION['visits'] = 0; }
else {
        $_SESSION['visits']++; }
```

the variable **visits** attached to the session

details at php.net/manual/en/book.session.php

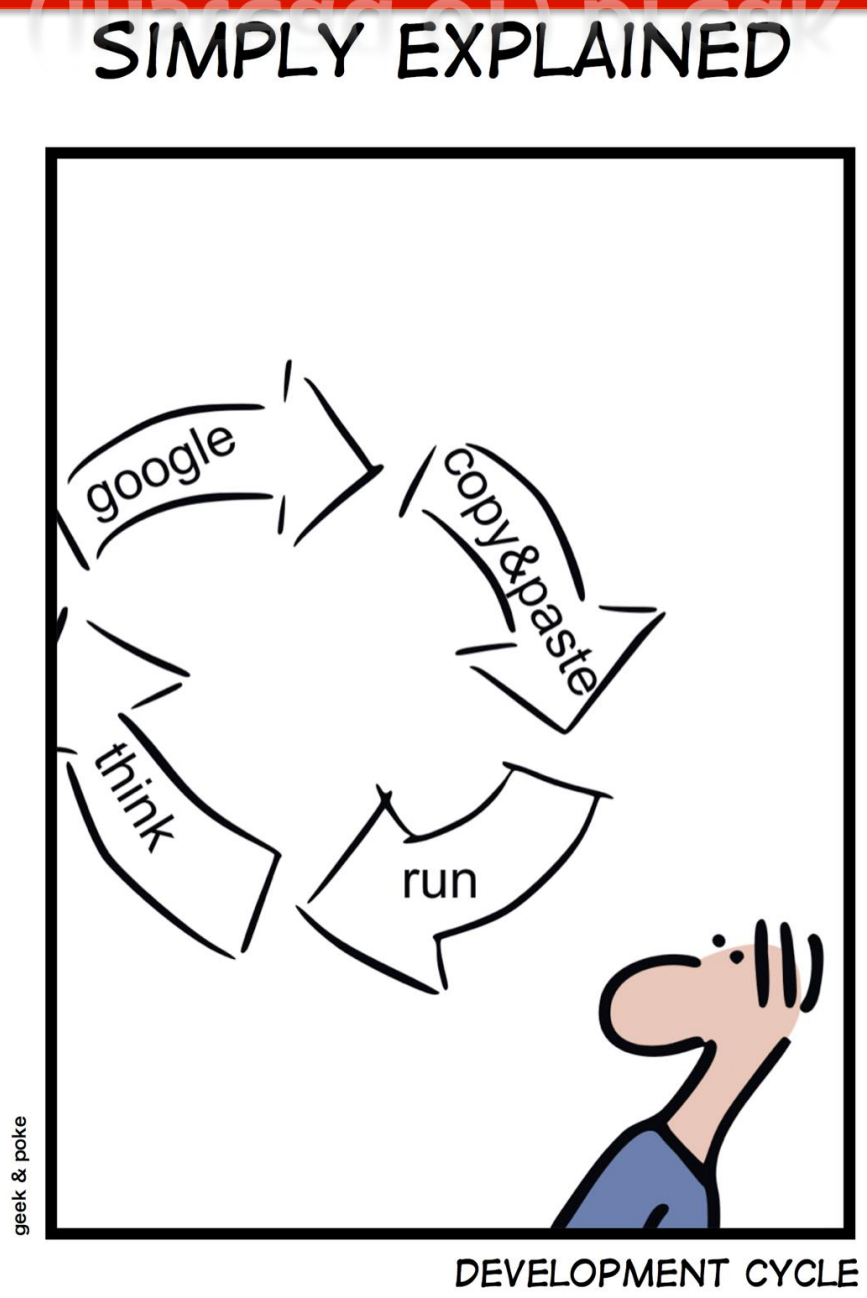
php: web sessions

Session management: SessionHandler class

```
SessionHandler implements SessionHandlerInterface {
  public bool   open ( string $save_path , string $session_name )
  public string create_sid ( void )
  public string read ( string $session_id )
  public bool   write ( string $session_id , string $session_data )
  public bool   gc ( int $maxlifetime )
  public bool   destroy ( string $session_id )
  public bool   close ( void )
}
```

php.net/manual/en/book.session.php

(instead of) break



How can databases be accessed from PHP?

php: db

Native support for multiple database servers/technologies:

MongoDB – classes: **MongoDB MongoClient MongoCursor**
MySQL / MariaDB – **mysqli** class
Oracle –**oci_*()** functions + **OCILob** class
PostgreSQL – **pg_*()** functions + **PgSql** class
SQLite – **SQLite3** class

connections can be persistent

php: db – mysql

Functions/methods to access MySQL/MariaDB

connecting to server: **mysql_connect ()**, **mysql_pconnect ()**

selecting (using) a database: **mysql_select_db ()**

executing a SQL statement: **mysql_query ()**

error reporting: **mysql_errno ()**, **mysql_error ()**

getting results into an array: **mysql_fetch_array ()**

many others…

in the present, **a deprecated approach** – eliminated in PHP 7+

php: db – mysqli extension

Aim: easy and flexible access to MySQL/MariaDB from PHP5+ programs

procedural or object-oriented approach
facilitates code maintenance

compatible with MySQL API

ensures security and performance

available docs at **www.php.net/mysqli**

php: db – mysqli extension

Important methods:

initiating a connection to a MySQL server – **mysqli ( )**
issuing SQL statements – **query ( )**, **prepare ( )**, **execute ( )**
processing the response – **fetch ( )**, **fetch_assoc ( )**
closing connection – **close ( )**
etc.

php: db – example

Initially, we'll create a MySQL account to allow authenticated access from PHP programs to the students database:

```
(console)$ mysql -u root -p mysql
Enter password: **********
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Server version: 10.4.11-MariaDB Source distribution

MariaDB [mysql]> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE,
DROP ON students.* TO 'tux@localhost' IDENTIFIED BY 'p@rola'
WITH GRANT OPTION;
Query OK, 0 rows affected (0.001 sec)
```

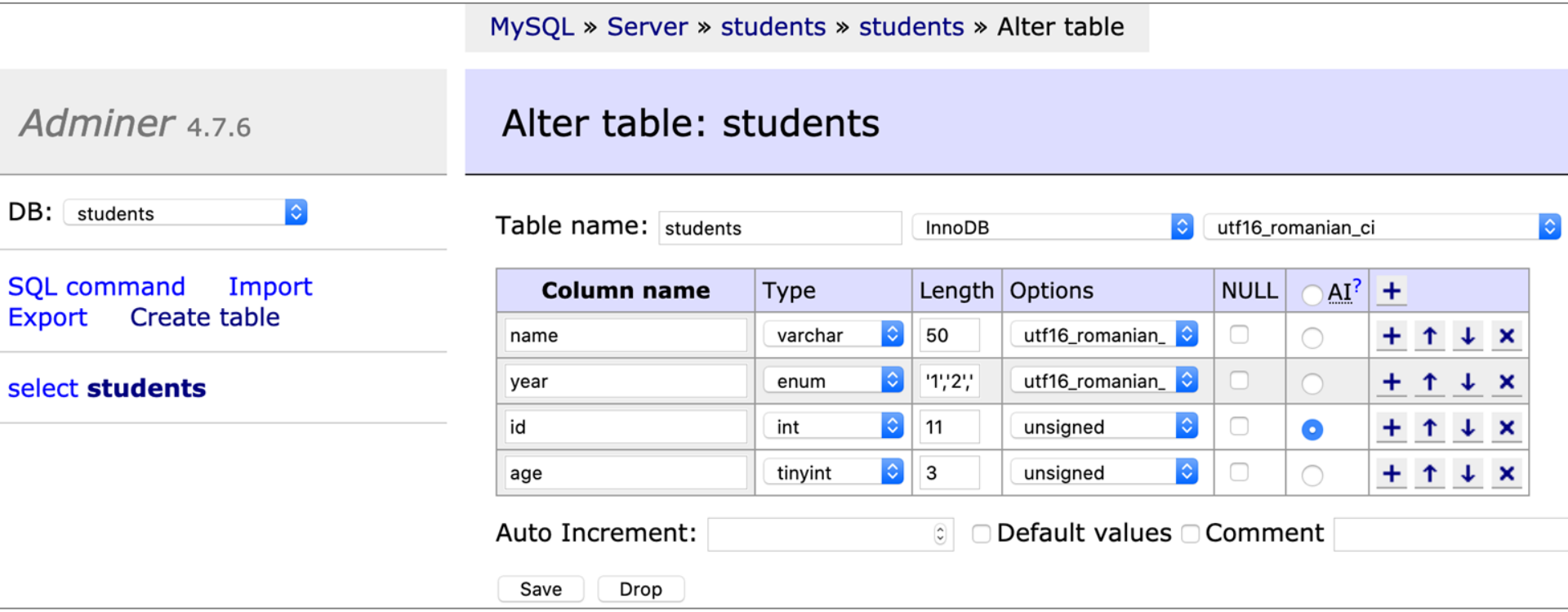
php: db – example

By using the mysql client in the command line or a Web administration application, we create the students table, having the structure:

```
+-------+---------------------+------+-----+---------+----------------+
| Field | Type                | Null | Key | Default | Extra          |
+-------+---------------------+------+-----+---------+----------------+
| name  | varchar(50)         | NO   |     | NULL    |                |
| year  | enum('1','2','3')   | NO   |     | NULL    |                |
| id    | int(11) unsigned    | NO   | PRI | NULL    | auto_increment |
| age   | tinyint(3) unsigned | NO   |     | NULL    |                |
+-------+---------------------+------+-----+---------+----------------+
```

for easy administration, use the Web tool

Adminer – **www.adminer.org**



visualising/altering the structure of the created table

alternative: phpMyAdmin – **www.phpmyadmin.net**

MySQL » Server » students » Select: students

Select: students

inserting records

Item has been inserted. 11:19:38 SQL command

```
INSERT INTO `students` (`name`, `year`, `id`, `age`)
VALUES ('Agarub Nibas Uilenroc', 2, '74', '26');
```

(0.004 s)

Edit

Select data Show structure Alter table

Select | Search | Sort | Limit 50

```
SELECT * FROM `students` LIMIT 50
```
(0.000 s) Edit

Modify	name	year	id	age
edit	Tuxy Pinguinnesscool	2	33	21
edit	Agarub Nibas Uilenroc	2	74	26

Whole result: 2 rows | Modify: Save | Selected (0): Edit Clone

Import

MySQL » Server » students » Export

Export: students

Output	open save gzip
Format	SQL CSV, CSV; TSV
Database	Routines Events
Tables	DROP+CREATE Auto Increment Triggers
Data	INSERT

Export

Tables	Data
students	~ 2

data export

SQL command

manual execution of SQL commands

```
SELECT name, age FROM students WHERE age > 21 and year = 2
```

name	age
Agarub Nibas Uilenroc	26

1 row (0.001 s) Edit, Explain, Export

id?	select_type?	table?	partitions?	type?	possible_keys?	key?	key_len?	ref?	rows?	Extra?
1	SIMPLE	students	*NULL*	ALL	*NULL*	*NULL*	*NULL*	*NULL*	2	Using where

```
SELECT name, age FROM students WHERE age > 21 and year = 2;
```

Execute Limit rows: Stop on error Show only errors

History

query history

Edit 11:28:50 `SELECT name, age FROM students WHERE age > 21 and year = 2;`
Edit 11:28:00 `SELECT year, name, age FROM students WHERE year=2 ORDER BY age;`
Edit 11:24:19 `SELECT name, age FROM students WHERE age > 21 and year = 2;`
Edit 11:23:30 `SELECT * FROM students WHERE age > 21 and year = 2 ;`

php: db – mysqli extension

```
// a mysqli object instantiation
$mysql = new mysqli ('localhost', 'tux', 'pa$$w0Rd', 'students');
if ( mysqli_connect_errno ()) {
   die ('Failed connection...');
}

// specifying & executing a query
if (!($res = $mysql->query ('select name, year from students'))) {
   die ('A query error occurred');
}
```

php: db – mysqli extension

```
// a mysqli object instantiation
$mysql = new mysqli ('localhost', 'tux', 'pa$$w0Rd', 'students');
if (mysqli_connect_errno()) {
   die ('Failed connection...');
}

// specifying & executing a query
if (!($res = $mysql->query('select name, year from students'))) {
   die ('A query error occurred');
}
```

The password is specified in clear text!

- **pay attention to security issues that may arise**

php: db – mysqli extension

```
// generating a numbered list of student-related data
// (HTML spaghetti code – not a recommended practice!)
echo ('<ol>');
// results will be available into an associative array
while ($row = $res->fetch_assoc ()) {
   // table column ≡ array key
   echo ('<li>Student ' . $row['name'] .
              ' is enrolled in ' . $row['year'] . ' year</li>');
}
echo ('</ol>');

// closing connection to MySQL/Maria DB server
$mysql->close ();
```

php: db

In practice, we'll adopt an abstraction layer in order to access a storage system

DBAL – DataBase Abstraction Layer

usual approach:
PDO (*PHP Data Objects*)

pragmatic aspects in the tutorial **phpdelusions.net/pdo**

```php
// connection data regarding MySQL/MariaDB database server
$host = '127.0.0.1';
$db   = 'students';
$user = 'tux';
$pass = 'pa$$w0Rd'; // important: clear text password
$charset = 'utf8';

// setting the DSN (Data Source Name)
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";

// options concerning the connection manner
$opt = [
    // errors will be reported as exceptions
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    // results will be stored by an associative array
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    // connection is persistent
    PDO::ATTR_PERSISTENT         => TRUE
];
```

php.net/manual/en/book.pdo.php

```php
// retrieve from the Web client the year of study (default: 2)
$year = $_REQUEST['year'] ? $_REQUEST['year'] : 2;

try {
    $pdo = new PDO ($dsn, $user, $pass, $opt); // instantiating a PDO object

    // preparing a parametrized SQL statement
    $sql = $pdo->prepare ('SELECT year, name, age FROM students
                           WHERE year=? ORDER BY age');

    if ($sql->execute ([$year])) {          // SQL statement can be executed?
      while ($row = $sql->fetch()) {        // getting each found record
        // ...and showing it (table column is a key of associative array)
        echo '<p>' . $row['name'] . ' is enrolled in ' . $row['year'] . ' year</p>';
      }
    }
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage(); // the message of occurred exception
};
```

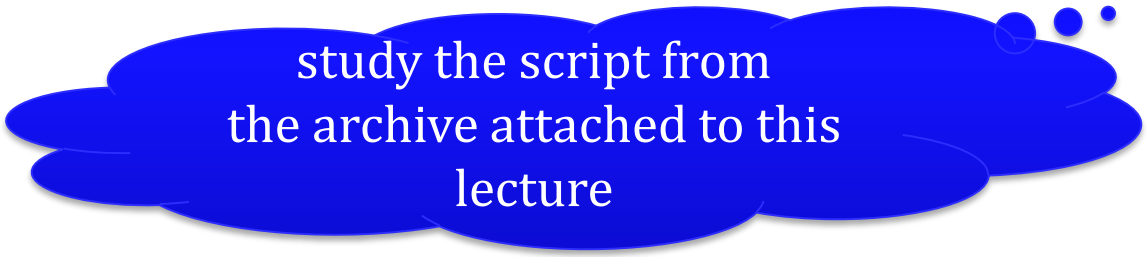
php: db – pdo extension

A possible result provided by the execution
of the PHP program invoked by using a URL as
https://profs.info.uaic.ro/~busaco/php/pdo-test.php?year=2

Tuxy Pinguinesscool is enrolled in 2 year

Grace Hopper is enrolled in 2 year

Margaret Hamilton is enrolled in 2 year



advanced

php: db

Usually, an ORM – Object-Relational Mapping
solution (component, library)
will be used on top of DBAL

examples:

Doctrine – **www.doctrine-project.org**

Propel – **propelorm.org**

RedBeanPHP – **redbeanphp.com**

Useful tools for Web developers?

advanced

tools: frameworks

Features:

MV* and various design patterns

database access (ORM, DAO, ActiveRecord,…)

input data validation and filtering

authentication + access control

cookie & session management

advanced

tools: frameworks

Features:

data presentation – templating

performance – *i.e.* caching

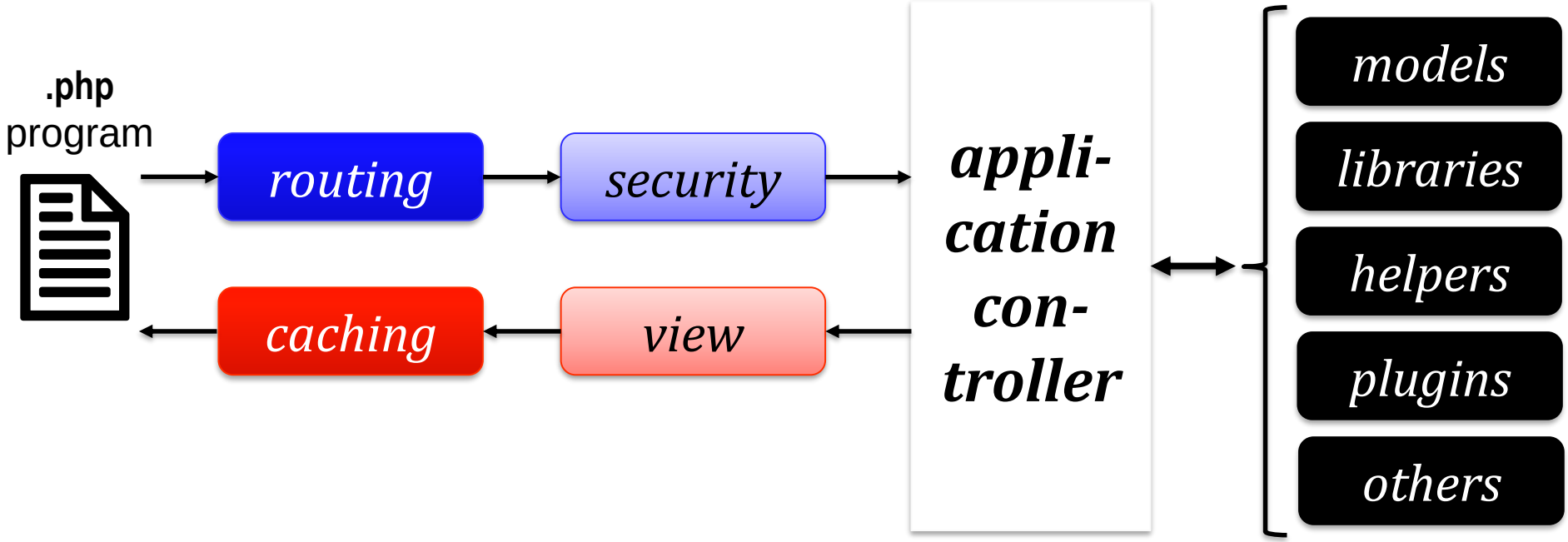
asynchronous data transfer (Ajax, WebSocket)

support for Web services and REST APIs

extensibility – *e.g.*, user-defined modules and managed with Composer tool

advanced

tools: frameworks



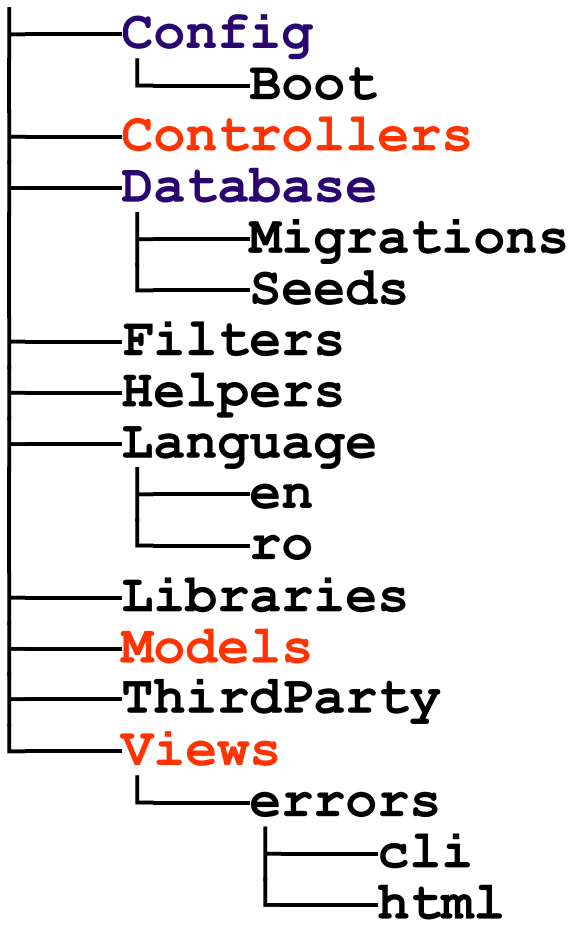
workflows performed by a framework

advanced

tools: frameworks

CakePHP – **cakephp.org**
CodeIgniter – **codeigniter.com**
FuelPHP – **fuelphp.com**
Laminas (ex-Zend Framework) – **docs.laminas.dev**
Laravel – **laravel.com**
Nette – **nette.org**
Phalcon – **phalcon.io**
PRADO – **www.pradoframework.net**
Symfony – **symfony.com** + Symfony Flex – **flex.symfony.com**
Yii – **www.yiiframework.com**

advanced



directory structure of
a Web application developed
by using a MVC framework

CodeIgniter 4

codeigniter.com/user_guide/

avansat

tools: frameworks

Some frameworks like Laravel can benefit from the facilities offered by a virtualization platform like Docker

packaging an (Web) application and its dependencies in a virtual container that can run on any operating system installed on a host (on premises) and/or configured by a vendor – usually, in cloud

laravel.com/docs/
laracasts.com

advanced

tools: micro-frameworks

A micro-framework represents
a minimalist Web framework

advanced

tools: micro-frameworks

A micro-framework represents
a minimalist Web framework

does not include complex features

often, is focused on a single aspect regarding
Web development – *e.g.*, creating an API, microservice,…

advanced

tools: micro-frameworks

Fat-Free – **fatfreeframework.com**
Flight – **flightphp.com**
Fresh Squeezed Limonade – **github.com/yesinteractive/fsl**
Leaf PHP – **leafphp.netlify.app**
Slim – **www.slimframework.com**
Siler – **siler.leocavalcante.dev**

advanced

tools: packages

Managing dependencies
between libraries and packages

Composer

getcomposer.org

www.phptherightway.com/#dependency_management

advanced

tools: packages

Meta-data + dependencies of other (versions of) packages are specified in the file **composer.json**

placed in the / root directory of the PHP project

according to the scheme **MAJOR.MINOR.PATCH**
(semantic versioning – **semver.org**),
dependency versions may have constraints specified via the symbols * < <= > >= ~ ^
details at **getcomposer.org/doc/articles/versions.md**

advanced

```
{
   "name": “org/package",
   "description": "...",
   "homepage": "http://package.org/",
   "minimum-stability": "stable",
   "version": "1.9.74",         // current version of the package
   "license": [ "LGPL-2.1-only", "GPL-3.0-or-later" ], // license
   "authors": [ { "name": "Tuxy", "role": "Developer" } ], // author(s)
   "require": {
      "php" : "^5.5 || >=7.4",     // PHP versions required to execute the code
      “namespace/otherpackage": "~2.0", // external dependency version
      "org/package2": "*"               // another package (any vers.)
   },
   "repositories": [      // download source
      { "type": "composer", "url": "http://packages.some.where/package" } ]
}
```

Composer will be used
(other values: **git**, **vcs**, **pear**, **package**)

advanced

tools: packages

Packages are encapsulated in PHAR (PHP Archive) and can be public/private

advanced

tools: packages

Packages are encapsulated in PHAR (PHP Archive) and can be public/private

php composer.phar command

where **command** can be:

install – local/global installation including dependencies

update – package update **remove** – package deletion

search – package search

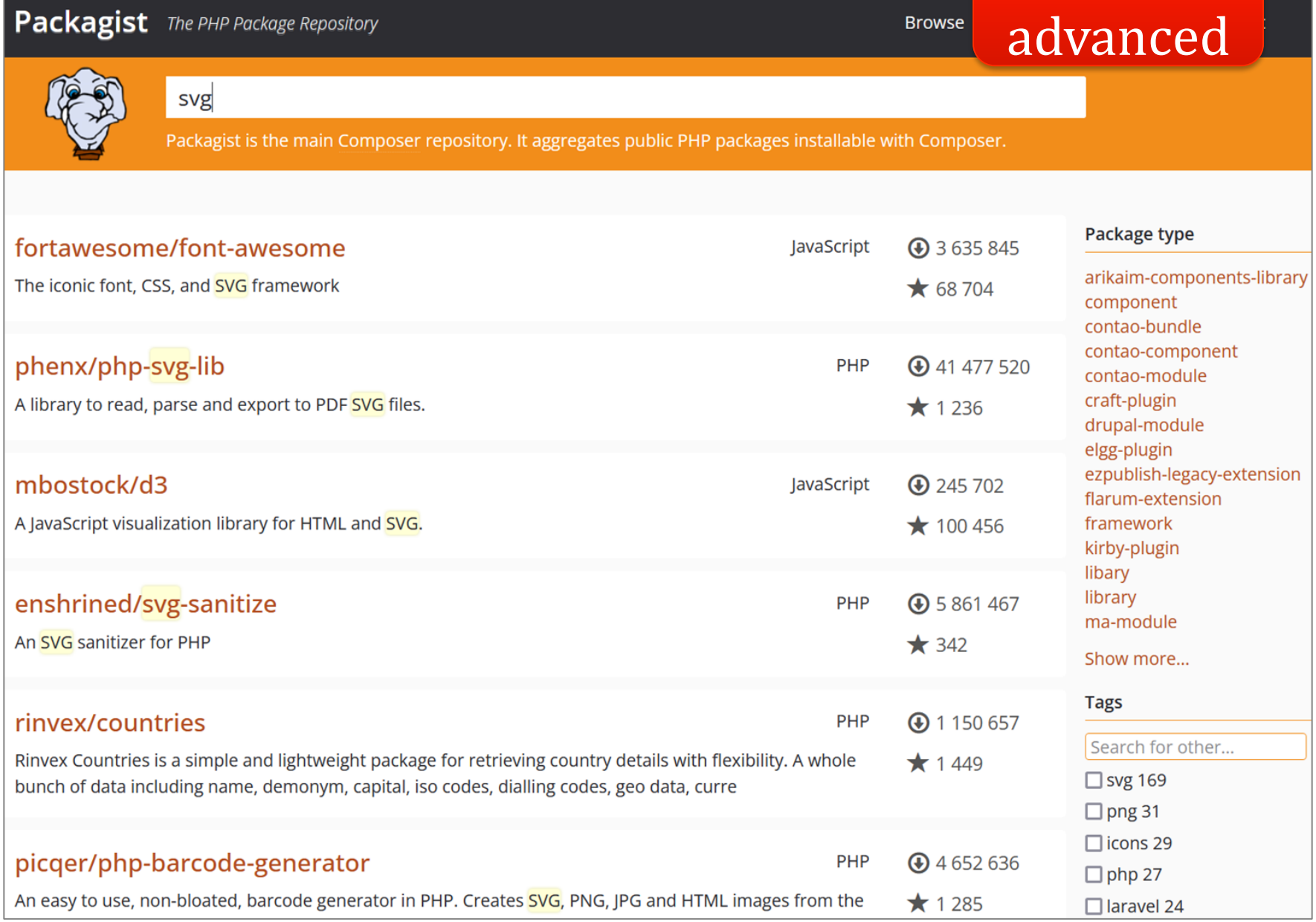
show – display all available packages

depends – display the packages that depend on a specific package

outdated – search for old packages

help – help information **diagnose** – diagnostic

...and others



public packages can be downloaded from the Web from Packagist – package repository: packagist.org

tools: packages

Dependency management
between libraries and packages

alternative – older:
PEAR (PHP Extension and Application Repository)
pear.php.net
+
extensions offered by third parties:
PECL (PHP Extension Community Library)
pecl.php.net

tools: development environments

Pre-configured Web development environments

Web server + PHP + database server(s) + tools

Apache + PHP7 + MySQL/MongoDB + Perl/Python + ...

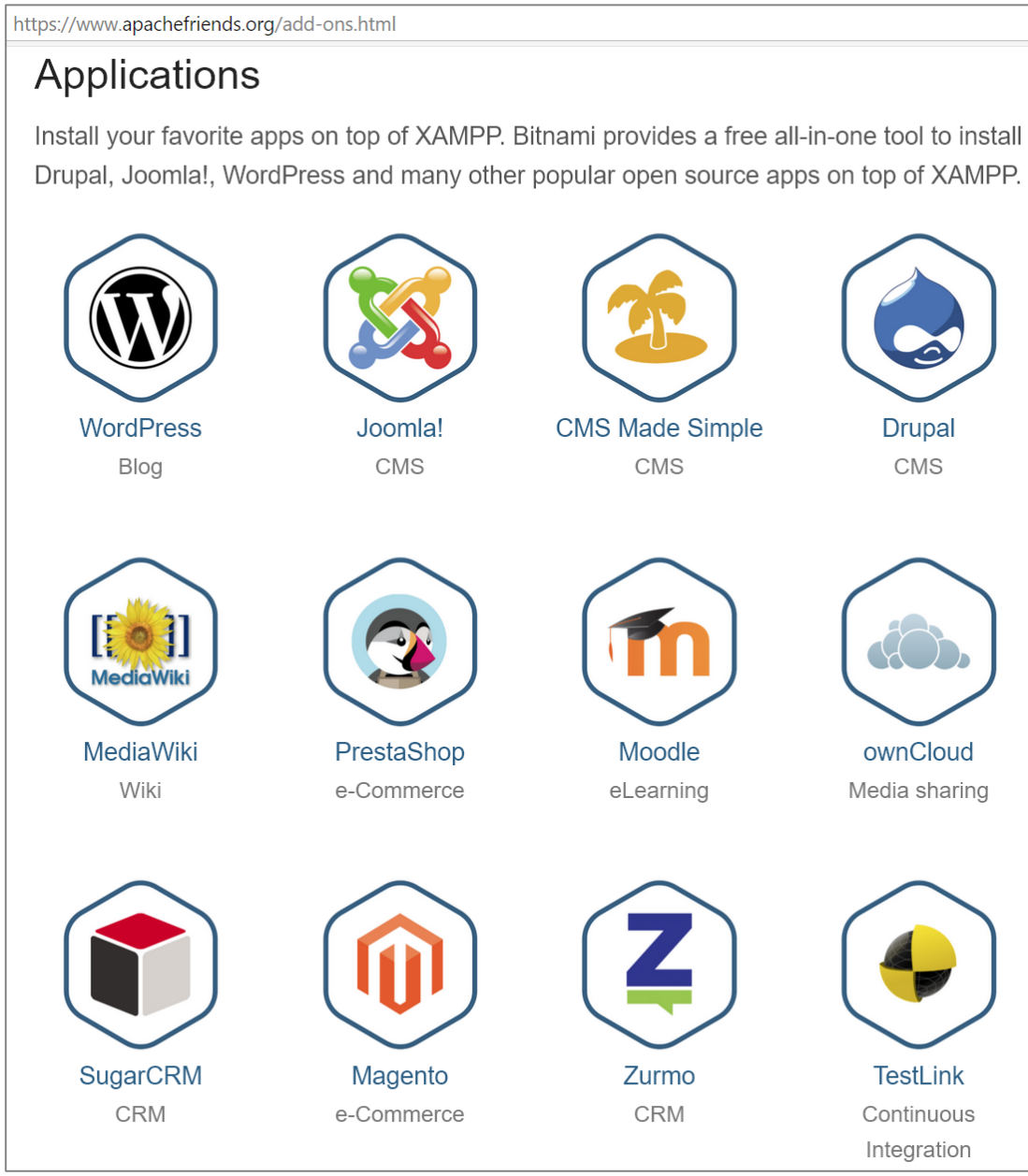
AMPPS

www.ampps.com

XAMPP

www.apachefriends.org

they also offer support for various configurations of Web systems (extensions)



WinNMP - Nginx MariaDB MongoDB Redis Php 7 development stack for Windows

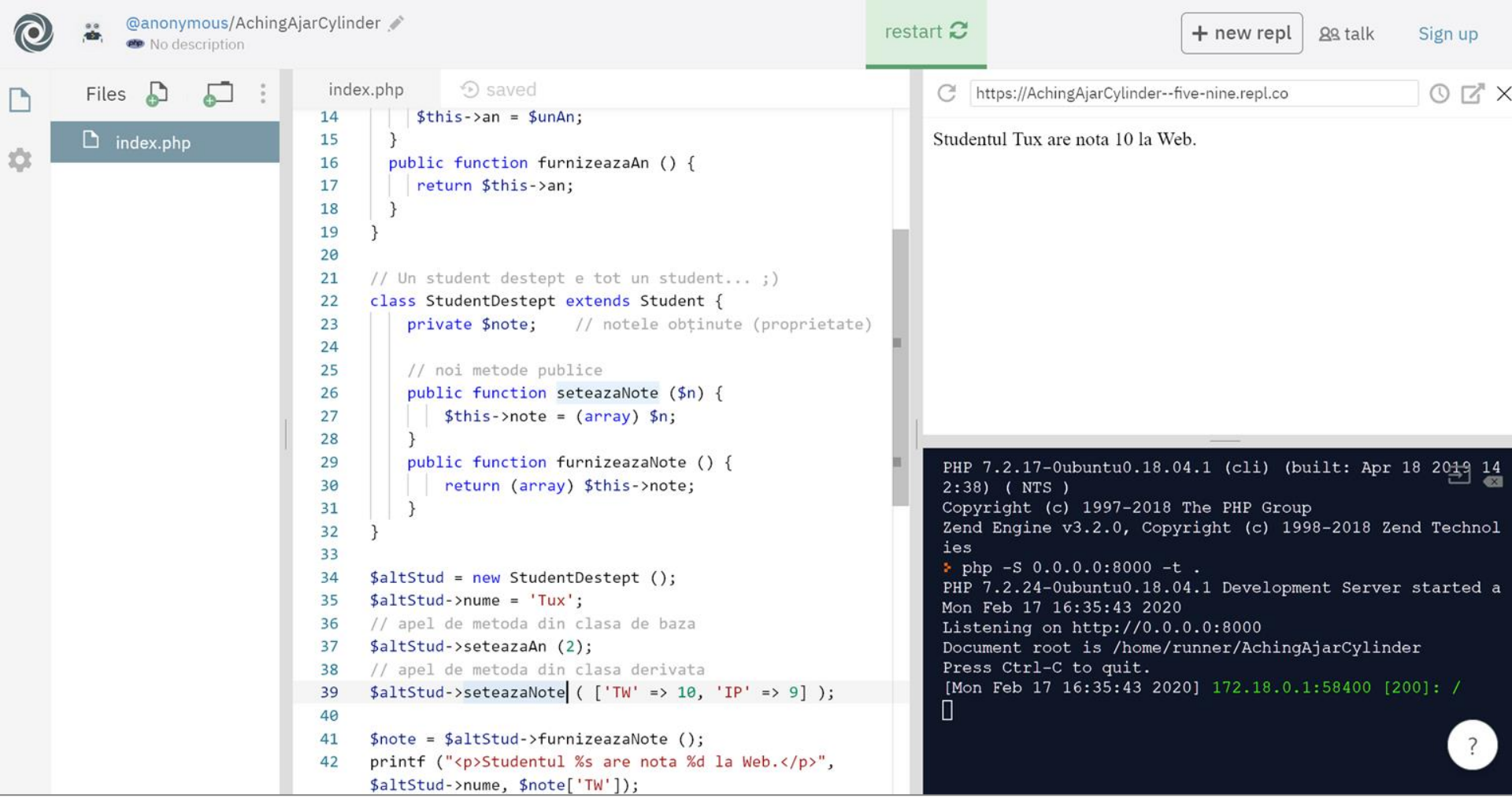
A lightweight, fast and stable server stack for developing php mysql applications on windows, based on the excellent webserver Nginx. A lighter alternative to XAMPP and WAMP, with Composer, Adminer, LetsEncrypt certificates, WinSCP, wp-cli, multiple PHP versions, projects and virtual servers.

Nginx, PHP7+, MariaDB, Redis + administration tools

WinNMP – winnmp.wtriple.com

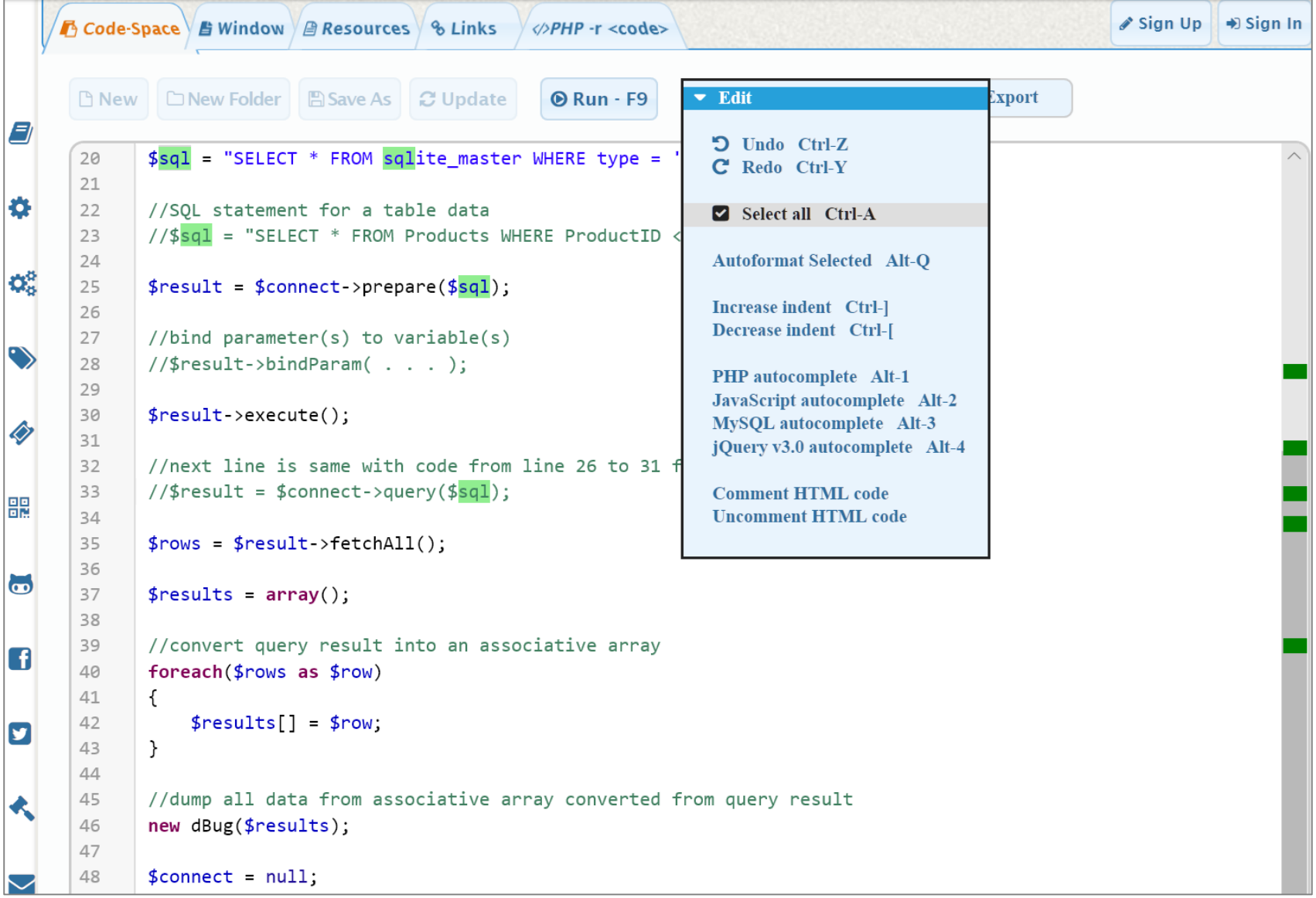
Current Package contains:

- Nginx 1.17.7 web server
- MariaDB 10.4.11 database server, mysql 5.5.5 replacement (32/64bit)
- MongoDB 4.2.2 document-based database (64bit)
- Redis 4.0 Cache/NoSql, memcached alternative (64bit)
- Php 5.6.40 & PHP 7.2.26 & PHP 7.3.13 & PHP 7.4.1 scripting language (32/64bit)
- XDebug, GeoIP, Gender, Mongodb PHP Extensions
- WinSCP SFTP client
- HTTPS using free LetsEncrypt certificates
- Composer dependency manager for php
- WP-CLI command-line interface for WordPress
- Adminer web based database manager



PHP (and not only) cloud based development environments available directly in the browser

AWS Cloud9, Koding, REPLit,...



online editing and execution of PHP programs

Ideone – ideone.com

PhpFiddle – phpfiddle.org

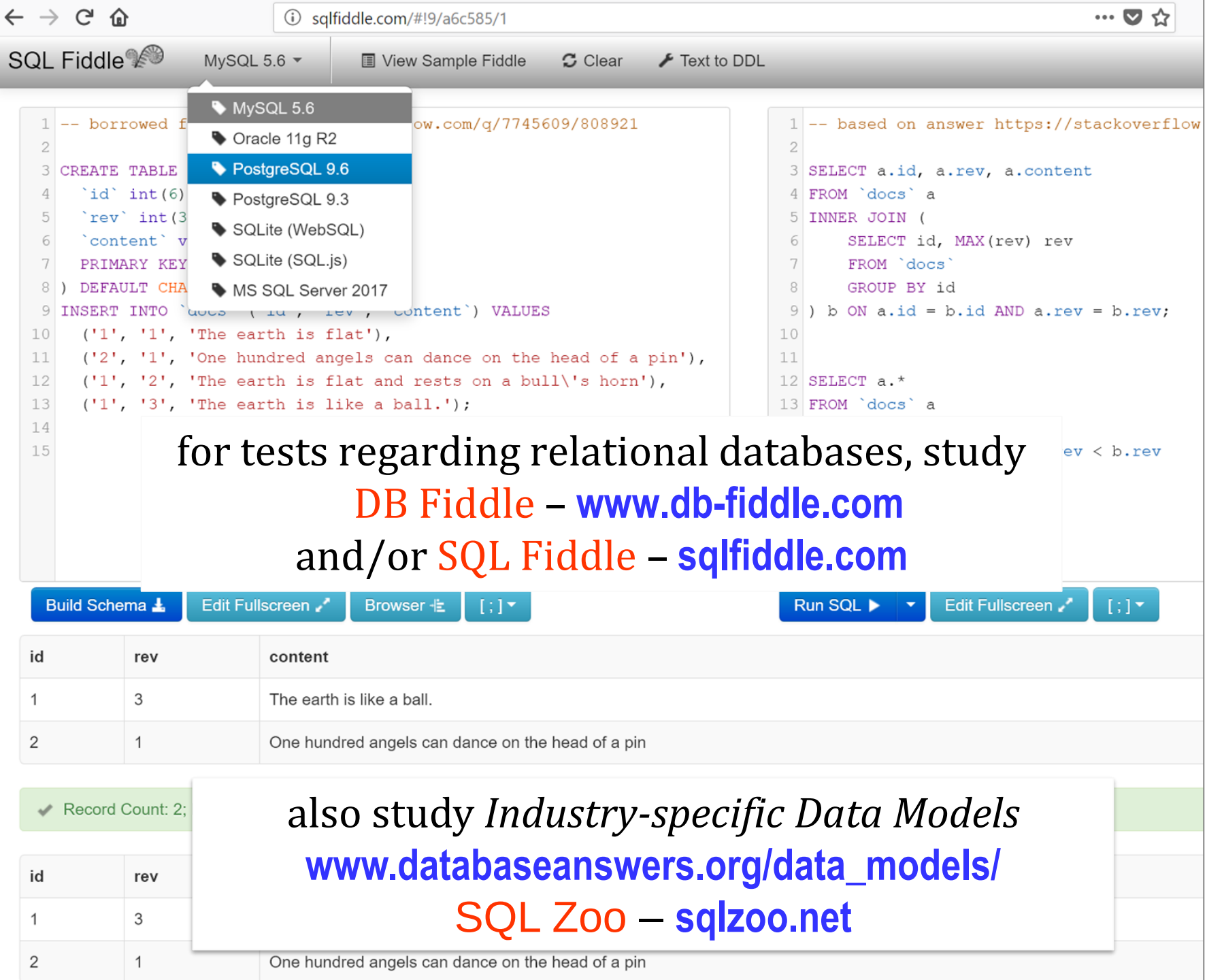
tools: online code execution

Online editing and execution of PHP programs

for quick code testing, consider:

PHP Sandbox – **sandbox.onlinephpfunctions.com**

PHPTester – **phptester.net**


sqlfiddle.com/#!9/a6c585/1
SQL Fiddle
MySQL 5.6
View Sample Fiddle
Clear
Text to DDL
MySQL 5.6
Oracle 11g R2
PostgreSQL 9.6
PostgreSQL 9.3
SQLite (WebSQL)
SQLite (SQL.js)
MS SQL Server 2017
-- borrowed f
ow.com/q/7745609/808921
CREATE TABLE
`id` int(6)
`rev` int(3
`content` v
PRIMARY KEY
) DEFAULT CHA
INSERT INTO
content`) VALUES
('1', '1', 'The earth is flat'),
('2', '1', 'One hundred angels can dance on the head of a pin'),
('1', '2', 'The earth is flat and rests on a bull\'s horn'),
('1', '3', 'The earth is like a ball.');
-- based on answer https://stackoverflow
SELECT a.id, a.rev, a.content
FROM `docs` a
INNER JOIN (
SELECT id, MAX(rev) rev
FROM `docs`
GROUP BY id
) b ON a.id = b.id AND a.rev = b.rev;
SELECT a.*
FROM `docs` a
ev < b.rev
for tests regarding relational databases, study
DB Fiddle – www.db-fiddle.com
and/or SQL Fiddle – sqlfiddle.com
Build Schema
Edit Fullscreen
Browser
[;]
Run SQL
Edit Fullscreen
[;]
id
rev
content
1
3
The earth is like a ball.
2
1
One hundred angels can dance on the head of a pin
Record Count: 2;
also study Industry-specific Data Models
www.databaseanswers.org/data_models/
SQL Zoo – sqlzoo.net
id
rev
1
3
2
1
One hundred angels can dance on the head of a pin

tools: documenting

Automatic documentation generators

Daux – **daux.io**
phpDocumentor – **www.phpdoc.org**
phpDox – **phpdox.de**

github.com/ziadoz/awesome-php#documentation

tools: documenting

Special markups in PHP comments:

@author
@category
@version
@copyright
@license
@see
@todo
@since
@deprecated

@var
@global
@method
@package
@subpackage
@param
@return
@throws
@inheritdoc

@example
@source
@uses
@used-by
@link
@internal
@property
@property-read
@property-write

docs.phpdoc.org/references/phpdoc/

tools: code analysis

PHP source-code static analysis

for discovering various programming bugs, verifying the conformance to various coding standards, automatic correction (fixers), determining code metrics: complexity, number of lines,…

github.com/exakat/php-static-analysis-tools

advanced

tools: code analysis

PHP Standards Recommendations – **www.php-fig.org/psr/**
written by PHP FIG (*Framework Interop Group*)

PSR1: *Basic Coding Standard*, PSR3: *Logger Interface*,
PSR4: *Autoloading*, PSR6: *Caching Interface*,
PSR7: *HTTP Message Interface*, PSR11: *Container Interface*,
PSR12: *Extended Coding Style Guide*,
PSR13: *Hypermedia Links*, PSR14: *Event Dispatcher*,
PSR15: *HTTP Handlers*,…

tools: phan, PHP_CodeSniffer, PHP-CS-Fixer, Rector

tools: code analysis

PSR12: *Extended Coding Style Guide* – example rules:

All PHP files MUST use the Unix LF (linefeed) line ending only.
The closing **?>** *tag MUST be omitted from files containing only PHP.*
Lines SHOULD NOT be longer than 80 characters.
There MUST NOT be more than one statement per line.
All PHP reserved keywords and types MUST be in lower case.
Visibility MUST be declared on all properties and methods.
There MUST be one space after the control structure keyword.

www.php-fig.org/psr/psr-12/

ERROR	TRUE, FALSE and NULL must be lowercase; expected "false" but found "FALSE"	Line 55, column 9
ERROR	Opening parenthesis of a multi-line function call must be the last content on the line	Line 55, column 26
ERROR	Expected at least 1 space before "=>"; 0 found	Line 59, column 17
ERROR	Multi-line function call not indented correctly; expected 4 spaces but found 10	Line 60, column 1
ERROR	Only one argument is allowed per line in a multi-line function call	Line 60, column 39
ERROR	Closing parenthesis of a multi-line function call must be on a line by itself	Line 60, column 43
ERROR	The closing parenthesis of a multi-line control structure must be on the line after the last expression	Line 60, column 44

reports provided by **phptools.online/php-checker**

ERROR	Whitespace found at end of line	Line 67,

advanced

tools: testing

atoum – atoum.org
Codeception – codeception.com
ParaTest (parallel testing) – github.com/brianium/paratest
Peridot – peridot-php.github.io
PHPUnit – phpunit.de
SimpleTest – github.com/simpletest

others at github.com/ziadoz/awesome-php#testing

advanced

tools: testing

Profiling – analyzing and reporting
slow-running code snippets

PhpBench – **github.com/phpbench/phpbench**
Tracy – **tracy.nette.org**
Xdebug extension for PHP – **xdebug.org/docs/profiler**
XHGui (bazat pe XHProf) – **github.com/perftools/xhgui**
Z-Ray – **www.zend.com/en/products/server/z-ray**

github.com/ziadoz/awesome-php#debugging-and-profiling

advanced

tools: continuous integration

CircleCI – **circleci.com**
PHPCI – **www.phptesting.org**
Sismo – **sismo.sensiolabs.org**

github.com/ziadoz/awesome-php#continuous-integration

advanced

extensions

Hack (Facebook, from 2014)
a programming language for HHVM, extending PHP

goal: increasing the Web developer productivity

features: explicit data types (type annotations), generics, λ expressions, asynchronous programming (async), and others

hacklang.org

major critics

many years: lack of language formal specification
now: github.com/php/php-langspec

inconsistency – *e.g.*, foreach, predefined function names

lack of native support for Unicode (except PHP 7+)

lack of native support for multi-threading,
but possible via extensions like pthreads et al.

PHP Sadness

Function naming (underscores)

« Previous Sadness | Sadness Index | Next Sadness »

Even between similar/related functions, some use underscores between words, while others do not. Here are some examples:

get: php gettype | php get_class

str: php str_ireplace | php str_pad | php str_repeat | php str_replace | php str_shuffle | php str_split | php str_word_count | php strcasecmp | php strchr | php strcmp | php strcoll | php strcspn

encode: php base64_encode | php quoted_printable_encode | php session_encode | php rawurlencode | php urlencode | php gzencode

php: php php_uname | php php_sapi_name | php php_logo_guid | php phpinfo | php phpcredits | php phpversion

PHP Sadness – phpsadness.com

Significance: Consistency

Language consistency is very important for developer efficiency. Every inconsistent language feature means that developers have one more thing to remember, one more reason to rely on the documentation, or one more situation that breaks their focus. A consistent language lets developers create habits and expectations that work throughout the language, learn the language much more quickly, more easily locate errors, and have fewer things to keep track of at once.

major critics

In PHP, we can easily create applications that “adopt” the *Spaghetti Code* anti-pattern

```
<h1>My Users <a class="btn btn-primary" href="new_user.php" role="button">New User</a></h1>
<table class="table table-condensed">
    <tr>
        <th>Id</th>
        <th>Name</th>
        <th>Age</th>
        <th>Email</th>
        <th>Action</th>
    </tr>
    <?php
    // Get all users
    $stmt = $dbh->prepare("SELECT * FROM users");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    if ($stmt->execute()) {
        while ($row = $stmt->fetch()) {
            ?>
            <tr>
                <td><?php echo $row['id']; ?></td>
                <td><?php echo $row['name']; ?></td>
                <td><?php echo $row['age']; ?></td>
                <td><?php echo $row['email']; ?></td>
                <td>
                    <div class="btn-group" role="group" aria-label="...">
                        <a href="edit.php?id=<?php echo $row['id']; ?>" class="btn btn-default btn-sm">Edit</a>
                        <a href="index.php?delete=<?php echo $row['id']; ?>" class="btn btn-default btn-sm">Delete</a>
                    </div>
                </td>
            </tr>
            <?php
```

HTML Web interface

PHP for data access and processing

PHP for data presentation

github.com/hanafiah/masterphp/tree/master/spaghetti

case study: Wikipedia

Jimmy Wales & Larry Sanger (2001)

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

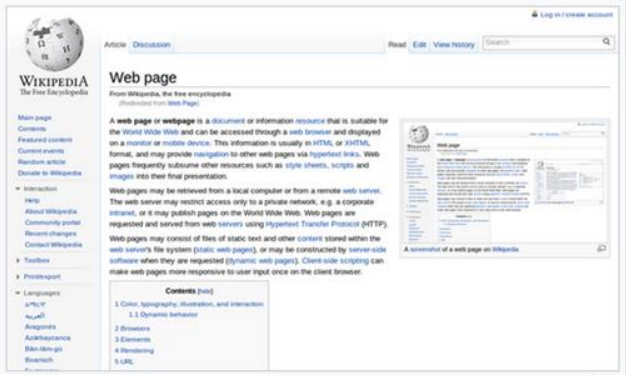
World Wide Web

From Wikipedia, the free encyclopedia

"WWW" and "The Web" redirect here. For other uses of WWW, see WWW (disambiguation). For uses of web, see Web (disambiguation).
For the first web software, see WorldWideWeb.
Not to be confused with the Internet.

The **World Wide Web** (**WWW**), commonly known as **the Web**, is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs, such as *https://www.example.com/*), which may be interlinked by hypertext, and are accessible over the Internet.[1][2] The resources of the WWW are transferred via the Hypertext Transfer Protocol (HTTP) and may be accessed by users by a software application called a *web browser* and are published by a software application called a *web server*.

British Scientist Tim Berners-Lee invented the World Wide Web in 1989. He wrote the first web browser in 1990 while employed at CERN near Geneva, Switzerland.[3][4] The browser was released outside CERN in 1991, first to other research institutions starting in January 1991 and then to the general public in August 1991. The World Wide Web has been central to the development of the Information Age and is the primary tool billions of people use to interact on the Internet.[5][6][7][8][9]



A web page can be displayed using a web browser. Web browsers often highlight and underline hypertext links and web pages can contain images.

case study: wikipedia

Aim: providing open content by using a group of collaborative Web applications – wikis

Wikipedia Foundation
also maintains Wikipedia, Wiktionary, Wikibooks, Wikiquote, Wikivoyage, Wikisource, Wikimedia Commons, Wikispecies, Wikinews, Wikiversity, Wikidata
en.wikipedia.org/wiki/Wikimedia_Foundation

case study: wikipedia

Complex version of the LAMP technology stack

MediaWiki - *wiki* system used for all services
implemented in PHP (7.4.3+ versions) + JavaScript
MariaDB (main storage solution – since 2013)
ImageMagick, DjVu, TeX, rsvg, ploticus, etc.
(for graphical content processing within MediaWiki)
Apache HTTP Server + NGINX (Web servers)

also provides an API to be used by Web developers:
www.mediawiki.org/wiki/API:Main_page

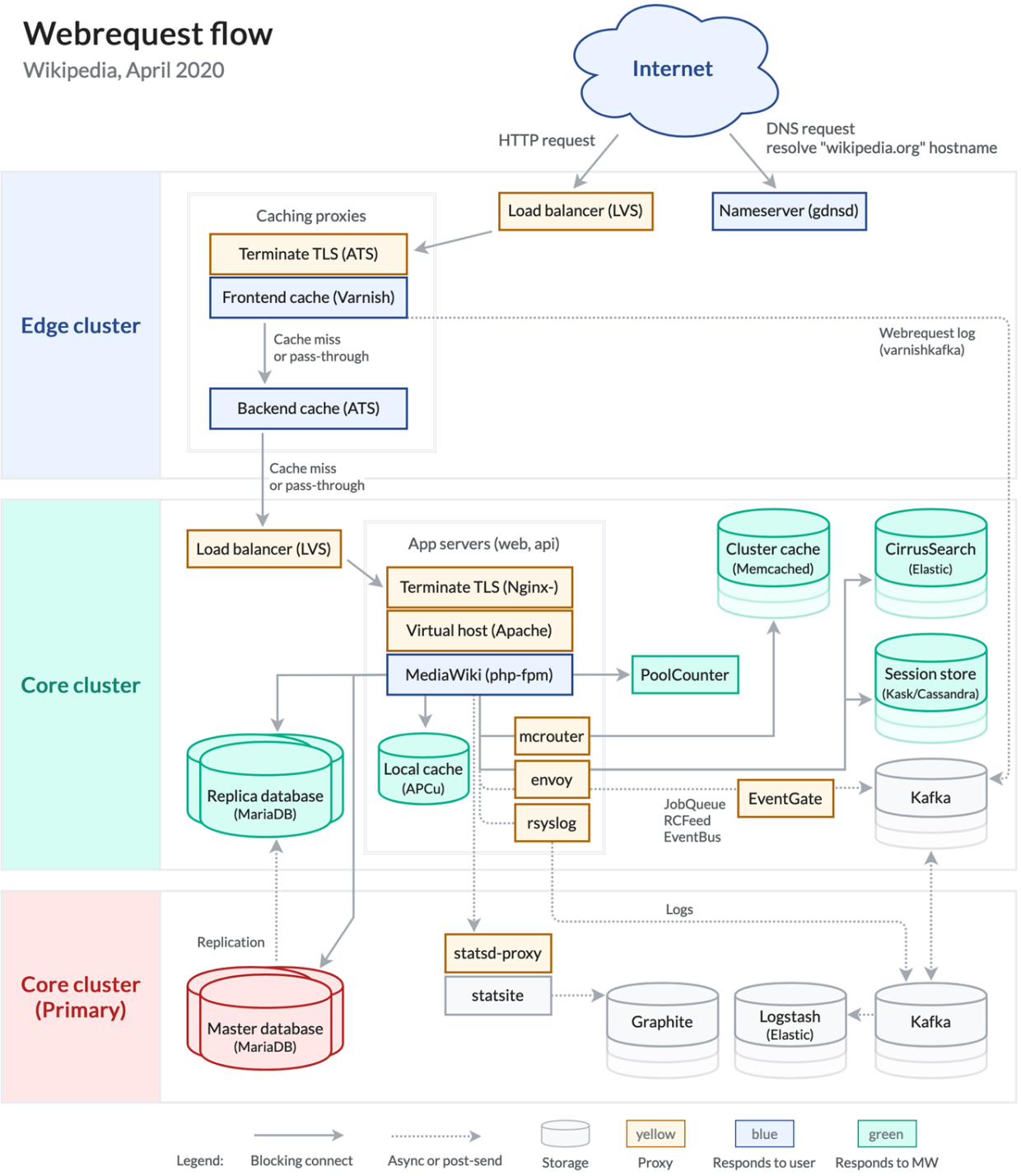
advanced

case study: wikipedia

PHP-FPM (*FastCGI Process Manager for PHP*)
Apache Traffic Server + Varnish (proxy + caching)
Memcached (caching for database queries)
Elasticsearch (textual search –Java implementation)
gdnsd (C++ solution for DNS)
Apache Kafka (event streaming – Java + Scala)
Linux Virtual Server (load balancing); PyBal (local monit.)
Debian GNU/Linux (OS)
Icinga + Grafana (monitoring the state of the systems)
Phabricator (bug tracking)

meta.wikimedia.org/wiki/Wikimedia_servers

advanced



main hosting:
USA, Virginia
secondary hosting:
USA, Texas
replication (caching):
Europe, Amsterdam
USA, San Francisco
Asia, Singapore

case studies

Various Web sites use Content Management Systems (CMS) developed in PHP

general:
Drupal, Joomla, WordPress etc.

What runs programmableweb.com?

CMS
- Drupal

Captcha
- reCAPTCHA

Miscellaneous
- Mouseflow
- Acquia

CDN
- Amazon Cloudfront

Analytics
- Google Analytics UA

Font Script
- Google Font API

Programming Language
- PHP 7.0.33

Sales and Marketing
- Klaviyo

Tag Managers

CMS can include extensions + themes

What runs geeksforgeeks.org?

CMS
- WordPress 5.2.2
- Theme: Iconic-one

Analytics
- Google Analytics UA

Font Script
- Google Font API

Cache
- WordPress Super Cache

Programming Language
- PHP 7.2.21

Widgets
- Google Plus
- Google Custom Search

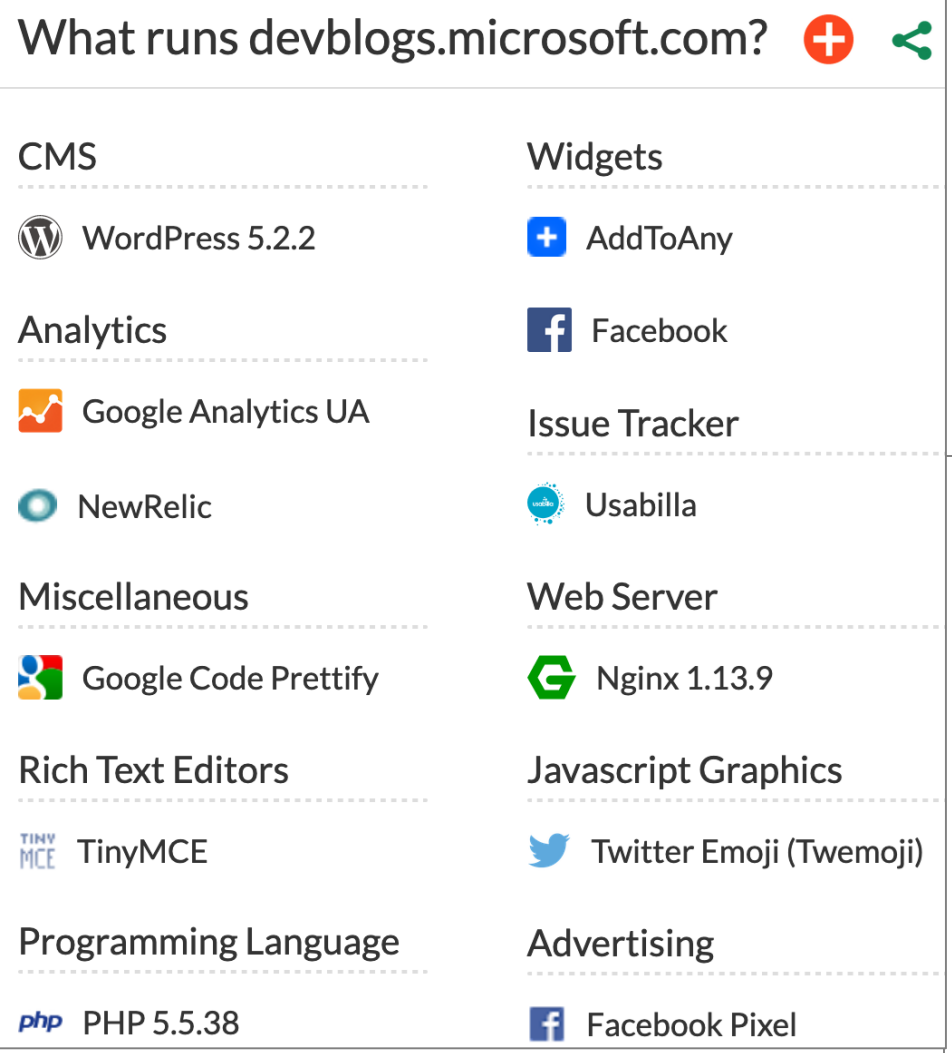
Comments and Reviews
- Disqus

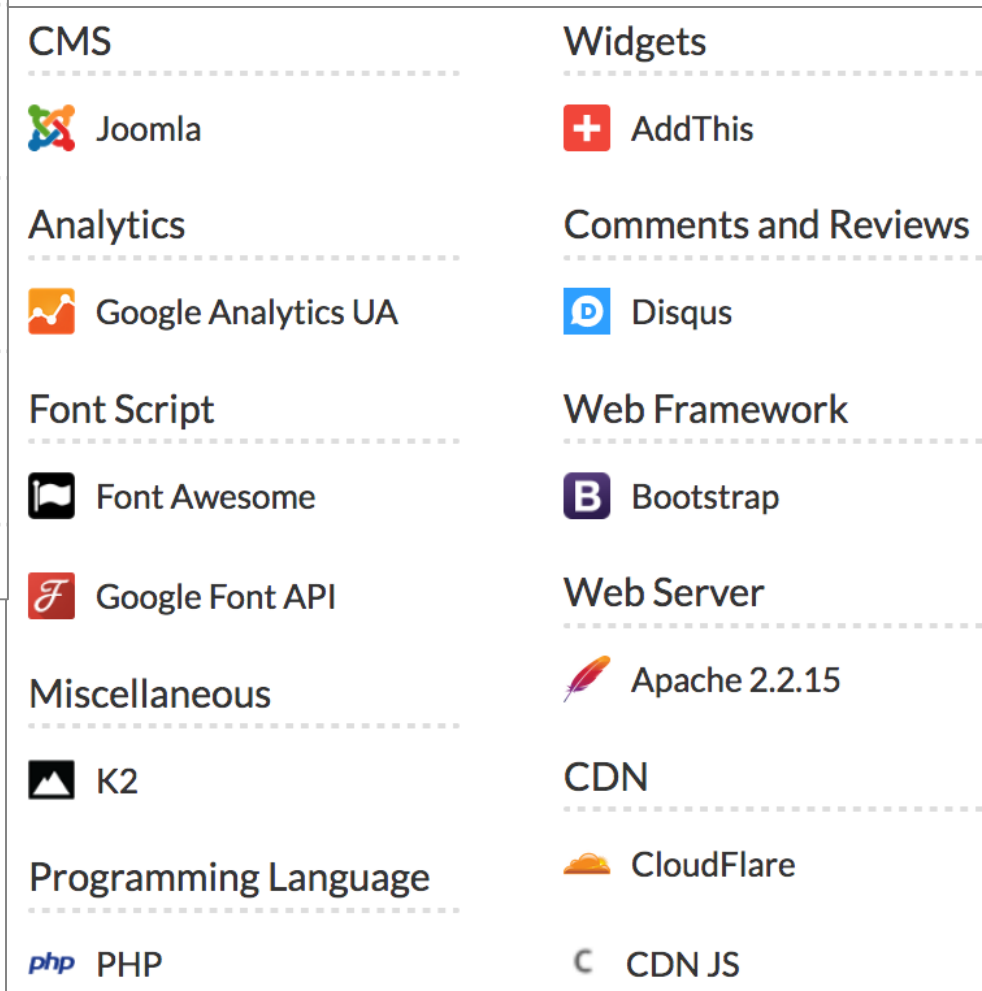
Web Server
- Apache 2.4.25

Javascript Graphics
- Twitter Emoji (Twemoji)

CDN
- Akamai

inspecting the technologies used by a Web application with the WhatRuns tool





inspecting the technologies used by a Web application with the WhatRuns tool

case studies

Various Web sites use Content Management Systems (CMS) developed in PHP

wiki:
DokuWiki, MediaWiki, pmWiki etc.

What runs wikidata.org?

Wiki

MediaWiki

Web Server

HHVM 3.18.6

Cache

Varnish

Programming Language

PHP 5.4.45

Dev Tools

HTML5 Shiv

Javascript Frameworks

jQuery

Font

Font: Sans Serif

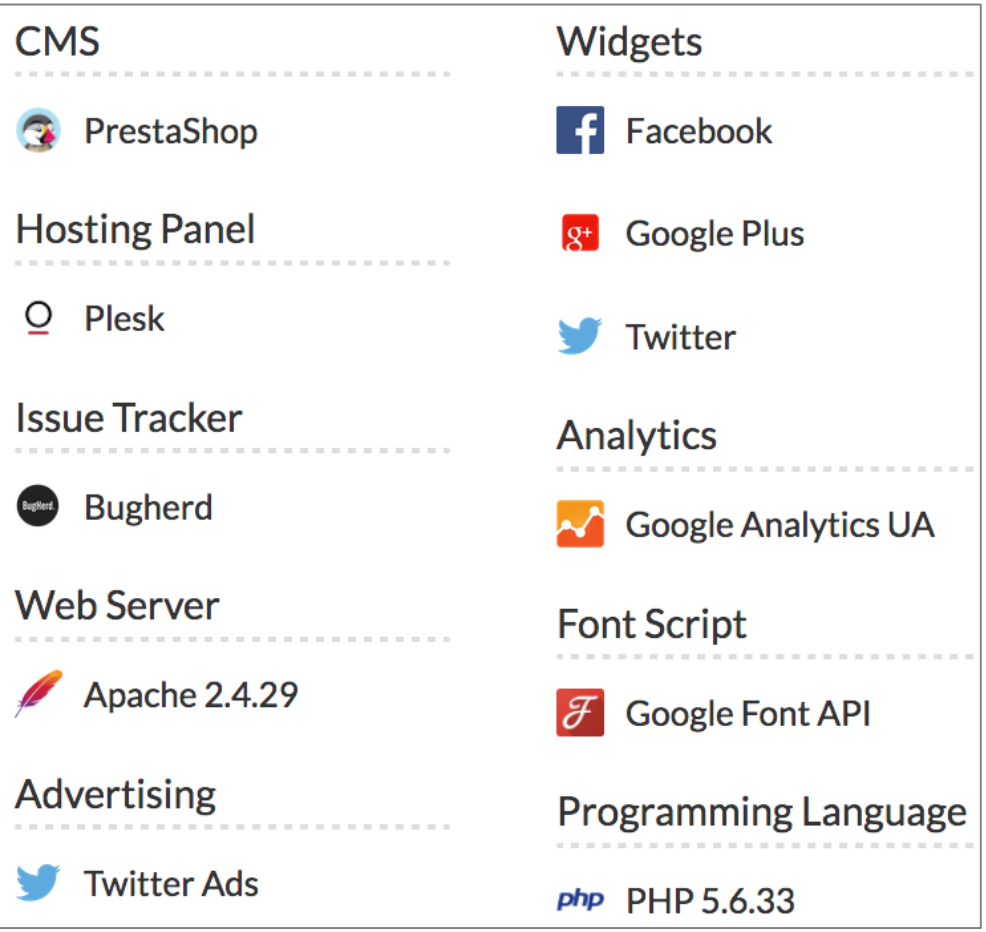
Font Family: Sans Serif

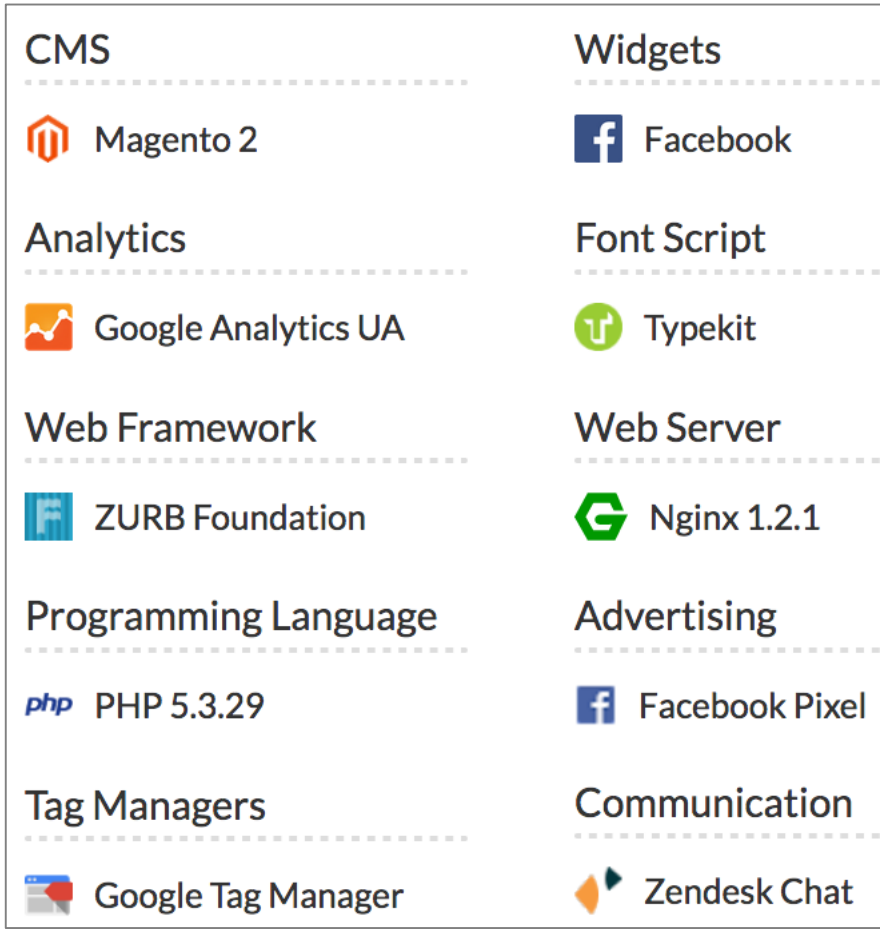
case studies

Various Web sites use Content Management Systems (CMS) developed in PHP

for e-commerce:
Magento, OpenCart, PrestaShop,…

case studies



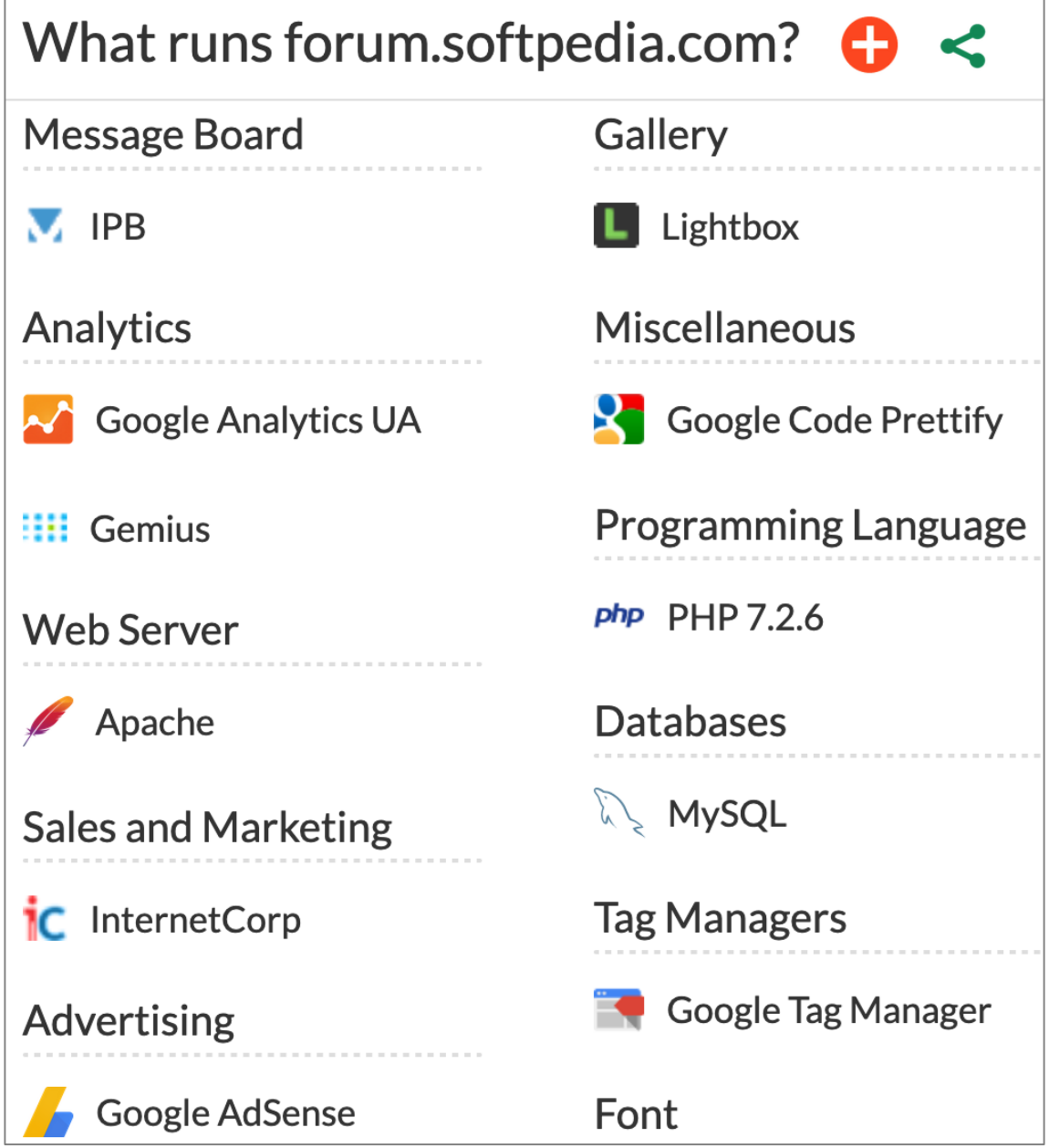


inspecting the technologies used by a Web application with the WhatRuns tool

case studies

Various Web sites use Content Management Systems (CMS) developed in PHP

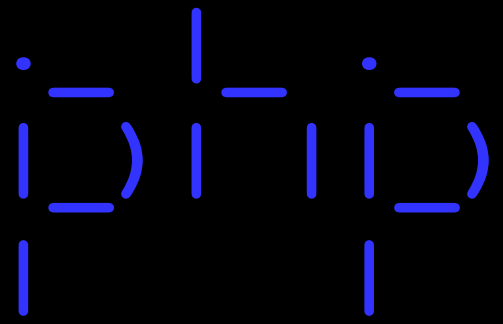
message board, Web forum:
bbPress, esoTalk, phpBB,…



inspecting the technologies used by a Web application with the WhatRuns tool

summary

PHP overview



characteristics, features, tools, examples

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<!-- pentru redarea datelor, se recurge la o foaie de stiluri CSS -->
<?xml-stylesheet type="text/css" href="game.css" ?>
<game>
    <title>Angry Profs</title>
    <platform type="tablet">Android</platform>
    <platform min-version="9" type="tablet">iOS</platform>
    <platform min-version="10">Windows</platform>
    <url>...</url>
    <player>
        <identity>
            <first-name>Sabin</first-name>
            <last-name>Buraga</last-name>
            <!-- ... -->
        </identity>
        <points>30374</points>
    </player>
</game>
```

next episode:

a data model for Web: XML family