Future Trends in Web Development

Simon Heimler
heimlersimon@gmail.com
Master of Applied Research Computer Science

Prof. Dr.-Ing. Christian Märtin
Faculty of Computer Science
University of Applied Sciences Augsburg

- INTRODUCTION
- TECHNOLOGICAL FOUNDATIONS OF THE FUTURE WEB
  - ECMASCRIPT HARMONY
  - WEB COMPONENTS
  - SEMANTIC WEB
- THE FUTURE WEB
  - WEB AS A PLATFORM
  - WEB OF APPLICATIONS
  - WEB OF SERVICES
- RESUME

Introduction

FULL DISCLAIMER

This is a subjective compilation of interesting trends.

Some trends are very likely to happen,
some are rather speculative!

INTRODUCTION

PREDICTING THE FUTURE?

- The future cannot be predicted, but thinking about the possibilities can be worthwhile!
- The seeds of the future grow in the present.
- Most technologies have decades of research and development behind when they reach market maturity.
- **Current trends** will shape the future. The difficulty is to find (and bet on) the most promising trends.

INTRODUCTION

HOW THE WEB EVOLVES

- The web is decentralized by nature.
- There is no central control where it is heading.
- It is increasingly developed from **bottom up**:
  - Users and developers set trends.
  - If they become widespread they will be standardized.
  - Example: HTML5 Living Standard.

INTRODUCTION

HOW THE WEB EVOLVES

- **PRO**: Democratic and “living” approach
- **PRO**: Standardized features have already been adopted.
- **CON**: This process can be messy and slow.
- **CON**: Developers can only use what currently works for the majority of the users.
- **CON:** Until then developers have to hack around the current limitations and browser differences.

Technological Foundations of the Future Web

ECMAScript Harmony

ECMASCRIPT HARMONY

TERMS AND CURRENT STATE

- ECMAScript (ES) is the official name of the JavaScript specification.
- ES Harmony refers to the two upcoming ES6 and ES7 standards.
- ES6 is expected to be officially released in June 2015
- Current browsers have already begun implementation
- Will take years until users and their browser have full support.

		Compilers				Desktop browsers														Server-ish				Mobile	
	37%	57%	55%	66%	30%	5%	21%	74%	45%	49%	58%	60%	47%	54%	9%	9%	24%	27%	9%	12%	0%	2%	3%	33%	9%	24%
Feature name	Current browser	Traceur	6to5 + polyfill	EJS	Closure Compiler	IE 10	IE 11	IE Technical Preview[1]	FF 31	FF 33	FF 34	FF 35	CH 38, OP 25[2]	CH 39, OP 26[2]	SF 6	SF 7.0	SF 7.1, SF 8	WK	OP 12	KQ 4.14[3]	RH	PH	Node	Node harmony[4]	iOS7	iOS8
proper tail calls (tail call optimisation)	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
arrow functions	0/9	7/9	7/9	7/9	6/9	0/9	0/9	8/9	7/9	7/9	7/9	7/9	3/9	4/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9	0/9
const	1/8	6/8	4/8	8/8	6/8	0/8	8/8	8/8	3/8	3/8	3/8	3/8	5/8	5/8	1/8	1/8	1/8	1/8	1/8	2/8	0/8	1/8	1/8	5/8	1/8	1/8
let	0/10	8/10	6/10	10/10	8/10	0/10	8/10	8/10	0/10	0/10	0/10	0/10	5/10	5/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0/10	4/10	0/10	0/10
default function parameters	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5	3/5	3/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5
rest parameters	No	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
spread (...) operator	0/8	8/8	8/8	6/8	2/8	0/8	0/8	4/8	6/8	6/8	6/8	6/8	0/8	0/8	0/8	0/8	2/8	2/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	2/8
class	0/8	7/8	7/8	6/8	6/8	0/8	0/8	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8
super	0/3	3/3	3/3	3/3	0/3	0/3	0/3	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
object literal extensions	0/3	3/3	3/3	3/3	3/3	0/3	0/3	3/3	0/3	1/3	3/3	3/3	0/3	1/3	0/3	0/3	1/3	1/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	1/3
for..of loops	4/4	4/4	4/4	3/4	3/4	0/4	0/4	4/4	3/4	3/4	3/4	3/4	4/4	4/4	0/4	0/4	1/4	1/4	0/4	0/4	0/4	0/4	0/4	1/4	0/4	1/4
generators	5/6	6/6	6/6	0/6	3/6	0/6	0/6	0/6	4/6	4/6	4/6	5/6	5/6	6/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	2/6	0/6	0/6
octal and binary literals	0/4	2/4	2/4	4/4	4/4	0/4	0/4	2/4	2/4	2/4	2/4	2/4	4/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	4/4	0/4	0/4
template strings	0/2	2/2	1/2	2/2	2/2	0/2	0/2	1/2	0/2	0/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
RegExp "y" and "u" flags	0/2	1/2	1/2	0/2	0/2	0/2	0/2	1/2	1/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
typed arrays	21/40	0/40	0/40	18/40	0/40	16/40	16/40	40/40	18/40	18/40	19/40	19/40	21/40	21/40	18/40	18/40	18/40	18/40	18/40	8/40	0/40	16/40	18/40	18/40	18/40	18/40
Map	11/11	10/11	11/11	9/11	0/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	10/11	11/11	0/11	0/11	9/11	9/11	0/11	0/11	0/11	0/11	0/11	6/11	0/11	9/11
Set	11/11	10/11	10/11	9/11	0/11	0/11	5/11	11/11	10/11	11/11	11/11	11/11	10/11	11/11	0/11	0/11	9/11	9/11	0/11	0/11	0/11	0/11	0/11	6/11	0/11	9/11
WeakMap	4/4	0/4	0/4	1/4	0/4	0/4	2/4	4/4	2/4	3/4	3/4	3/4	4/4	4/4	0/4	0/4	3/4	3/4	0/4	0/4	0/4	0/4	0/4	2/4	0/4	3/4
WeakSet	4/4	0/4	0/4	1/4	0/4	0/4	0/4	4/4	0/4	0/4	4/4	4/4	4/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	2/4	0/4	0/4
Proxy	0/17	0/17	0/17	8/17	0/17	0/17	0/17	14/17	11/17	12/17	13/17	13/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17
Reflect	0/14	0/14	0/14	14/14	0/14	0/14	0/14	13/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14	0/14
block-level function declaration[11]	No	No	No	No	Yes	No	Yes	Yes	No	No	No	No	Yes	Yes	No	No	No	No	No	No	No	No	No	Yes	No	No
destructuring	0/11	8/11	6/11	5/11	8/11	0/11	0/11	0/11	5/11	5/11	7/11	7/11	0/11	0/11	0/11	0/11	5/11	5/11	0/11	0/11	0/11	0/11	0/11	0/11	0/11	5/11
Promise	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No	No	No	Yes	No	Yes
Object static methods	3/4	3/4	0/4	4/4	0/4	0/4	1/4	3/4	2/4	2/4	3/4	3/4	3/4	3/4	0/4	0/4	0/4	0/4	0/4	1/4	0/4	0/4	1/4	3/4	0/4	0/4
function "name" property	2/16	0/16	0/16	0/16	0/16	0/16	0/16	0/16	3/16	3/16	4/16	4/16	2/16	3/16	3/16	3/16	3/16	3/16	2/16	3/16	3/16	3/16	2/16	2/16	3/16	3/16
Function.prototype.toMethod	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No
String static methods	0/2	2/2	2/2	2/2	0/2	0/2	0/2	2/2	1/2	1/2	2/2	2/2	1/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
String.prototype methods	1/6	4/6	4/6	4/6	0/6	0/6	0/6	5/6	5/6	5/6	5/6	5/6	5/6	5/6	0/6	0/6	0/6	2/6	0/6	0/6	0/6	0/6	0/6	3/6	0/6	0/6
Unicode code point escapes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Symbol	7/8	4/8	4/8	5/8	0/8	0/8	0/8	7/8	0/8	0/8	0/8	0/8	6/8	7/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	6/8	0/8	0/8
global symbol registry	Yes	No	No	Yes	No	No	No	Yes	No	No	No	No	Yes	Yes	No	No	No	No	No	No	No	No	No	Yes	No	No
well-known symbols	2/7	1/7	0/7	4/7	0/7	0/7	0/7	2/7	0/7	0/7	0/7	0/7	2/7	2/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7
RegExp.prototype methods	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4
Array static methods	0/2	2/2	2/2	2/2	0/2	0/2	0/2	2/2	1/2	2/2	2/2	2/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Array.prototype methods	3/8	6/8	7/8	7/8	0/8	0/8	0/8	8/8	5/8	6/8	6/8	6/8	6/8	6/8	0/8	0/8	5/8	5/8	0/8	0/8	0/8	0/8	0/8	6/8	0/8	5/8
Number properties	7/7	7/7	7/7	7/7	0/7	0/7	0/7	7/7	6/7	7/7	7/7	7/7	7/7	7/7	0/7	0/7	0/7	7/7	0/7	6/7	0/7	0/7	2/7	7/7	0/7	0/7
Math methods	17/17	0/17	17/17	17/17	0/17	0/17	0/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17	0/17	1/17	15/17	16/17	0/17	14/17	0/17	0/17	0/17	17/17	1/17	15/17
hoisted block-level function declaration	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No
__proto__ in object literals[19]	1/5	0/5	0/5	0/5	0/5	0/5	1/5	1/5	1/5	1/5	2/5	5/5	1/5	1/5	1/5	1/5	2/5	2/5	1/5	1/5	1/5	1/5	1/5	1/5	1/5	2/5
Object.prototype.__proto__	3/3	0/3	0/3	0/3	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	2/3	3/3	0/3	2/3	3/3	3/3	3/3
String.prototype HTML methods	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RegExp.prototype.compile	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Goals

ECMASCRIPT HARMONY

BACKWARD COMPATIBILITY

- It’s called Harmony for a reason.
- Completely backward compatible, does not introduce any breaking changes. (Remember the Python 3.x disaster)
- If old features need to be fixed, new features are introduced that can be used as a replacement
- **Examples**: `let` variable declaration, `for ... of` loop.

ECMASCRIPT HARMONY

COMPLEX APPLICATIONS

- JavaScript was never designed for writing big, complex application. It happened anyway.
- ES6 introduces a module system, that allow for better code organization and modularity
- **Examples**: New module system.

ECMASCRIPT HARMONY

CODE GENERATOR TARGET

- JavaScript is increasingly used as a target language of code generators.
- ES Harmony introduces some features that allow for better performance and machine-optimizations.
- **Examples**: Typed Arrays

ECMASCRIPT HARMONY

A BETTER PROGRAMMING EXPERIENCE

- Many new features are added that provide a better programming experience
- Some best practices went into core.
- Some features are only “syntactic sugar”, providing nicer ways to write the code
- **Examples**: Promises, class notation

ECMASCRIPT HARMONY

CONCLUSION

- ES Harmony makes JavaScript fit for modern application development.
- It is highly recommended to define and use only a smaller sub-set of the language.
- If a “Style Guide” is used, JavaScript is a minimalistic, but powerful language to work with.
- Until its fully available, developers can use transpilers (Google Traceur) or for add some features through shims.

Web Components

WEB COMPONENTS

TERMS AND CURRENT STATE

- Creation of modular, custom HTML elements that behave like native elements.
- Allowing developers to extend or alter the browser by giving them a more low-level access.

WEB COMPONENTS

EXTENDING THE BROWSER

- Web Components are a collection of several standards that are currently in W3C standardization process.
- Only Chrome and Firefox started implementation yet.

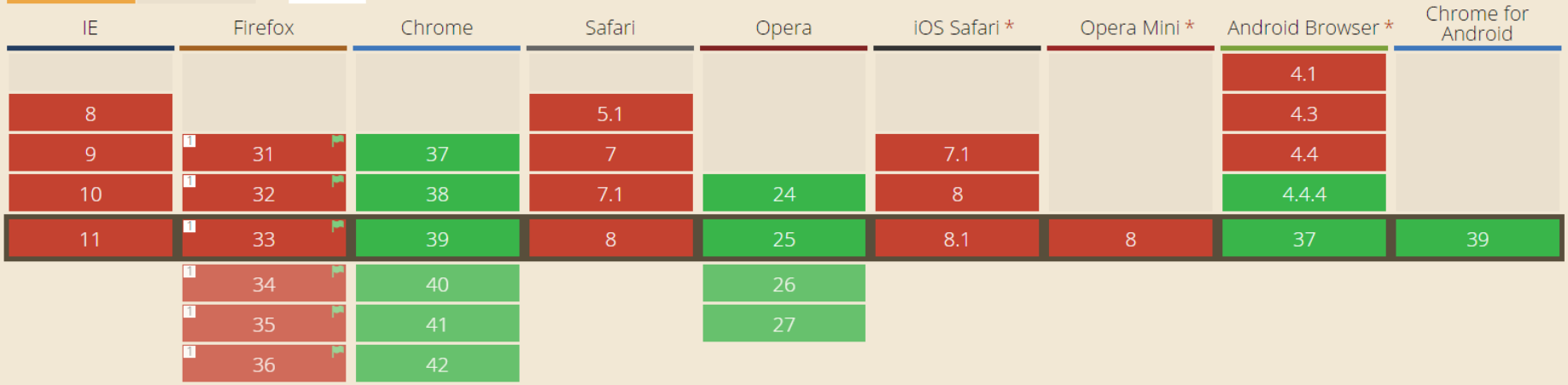
Custom Elements - WD

Germany 34.12%

Global 42.65%

Method of defining and using new types of DOM elements in a document.

Current aligned | Usage relative | Show all



Notes | Known issues (0) | Resources (9) | Feedback

Current IE status: Under Consideration

1 Enabled through the "dom.webcomponents.enabled" preference in about:config

Responsive Image Element

Example

WEB COMPONENTS

- Problem: Images need to be conditionally loaded in different resolutions, depending on the device.
- The current <img> element does not support this.
- Standardization of <picture> is on its way, but will take a while…
- Meanwhile: A lot of JavaScript hacks.

WEB COMPONENTS

- Developers can create new HTML elements, as a Web Component.
- It works and feels just like native elements.
- Web Components can be easily distributed, included and used.

Custom Elements

a web components gallery for modern web apps

Search in 634 elements...

Name	Description	Stars	Forks	Author
x-gif	A custom element for flexible GIF playback	1374	62	geelen
amazeui	Amaze UI，中国首个开源 HTML5 跨屏前端框架	840	253	allmobilize
voice-elements	Web Component wrapper to the Web Speech API, that allows you to do voice recognition and speech synthesis using Polymer	793	114	zenorocha
time-elements	Web component extensions to the standard <time> element.	758	19	github
github-card	A web component to show a card for your GitHub profile	520	42	pazguille
qr-code	Web Component for generating QR codes	276	27	educastellano
ReactiveElements	Allows to use React.js component as HTML element	170	9	PixelsCommander
prism-js	A Polymer element for syntax highlighting with Prism.js	116	5	addyosmani
chart-elements	Chart.js as Polymer Elements	111	15	robdodson

WEB COMPONENTS

```
1 <link rel="import" href="/WebComponent/x-picture.html">
```

- Web Components are imported through HTML Imports
- There is only one request needed:
  One file contains HTML, CSS and JavaScript

WEB COMPONENTS

```
<x-picture alt="description">
  <x-source src="/img/small.jpg"  media="(min-width: 200px)"></x-source>
  <x-source src="/img/medium.jpg" media="(min-width: 400px)"></x-source>
  <x-source src="/img/large.jpg"  media="(min-width: 800px)"></x-source>
</x-picture>
```

- The API is HTML, no programming knowledge needed.
- The behind-the-scenes complexity is hidden (ShadowDOM).
- Web Components have their own, protected scope, so they wont interfere with other elements and vice versa.

WEB COMPONENTS

CONCLUSION

- Are modular and easy to use.
- Web Components could change web-development and the way the web platform evolves on a more fundamental level.
- They empower “bottom up” development.
- New features and workarounds feel more native and less hacky.
- Special features don’t need to be standardized, since a Web Component is just fine.

Semantic Web

SEMANTIC WEB

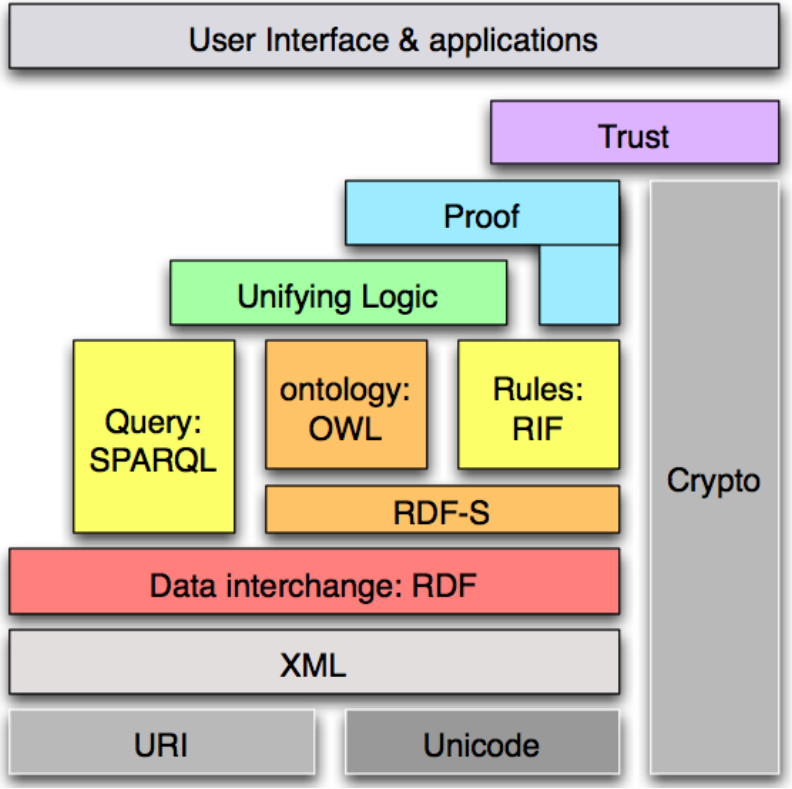
TERMS AND CURRENT STATE

- **Semantic Web**: Official name, used in academia and research
- **Linked Data**: Refers to the same technologies. Has a more pragmatic orientation. Focus on the data.
- Semantic Web is a combination of Web Technologies, AI, Data Science and Linguistics.

SEMANTIC WEB

TERMS AND CURRENT STATE

- Consists of a several standards and technologies, also called the **Semantic Web Stack.**
- Many of them are already W3C standardized or in the process of standardization.



SEMANTIC WEB

ADOPTION

- **Google, Microsoft, Yahoo and Yandex** started adding support. They are creating a global, standardized vocabulary for the web: schema.org.
- **Facebooks** Open Graph protocol is based on Semantic Web Technologies.
- However: Most Web-Devs and User don’t know about it.
- Adoption might be greatly accelerated with increasing SEO benefits and new applications.

SEMANTIC WEB

THE LIMITATIONS OF THE CURRENT WEB

- The current web is (mostly) optimized for human use.
- Machines have a hard time interpreting the content.
  AI technologies can guess the meaning with moderate success and quality.
- This makes automated data and information (re)usage between different websites complicated.

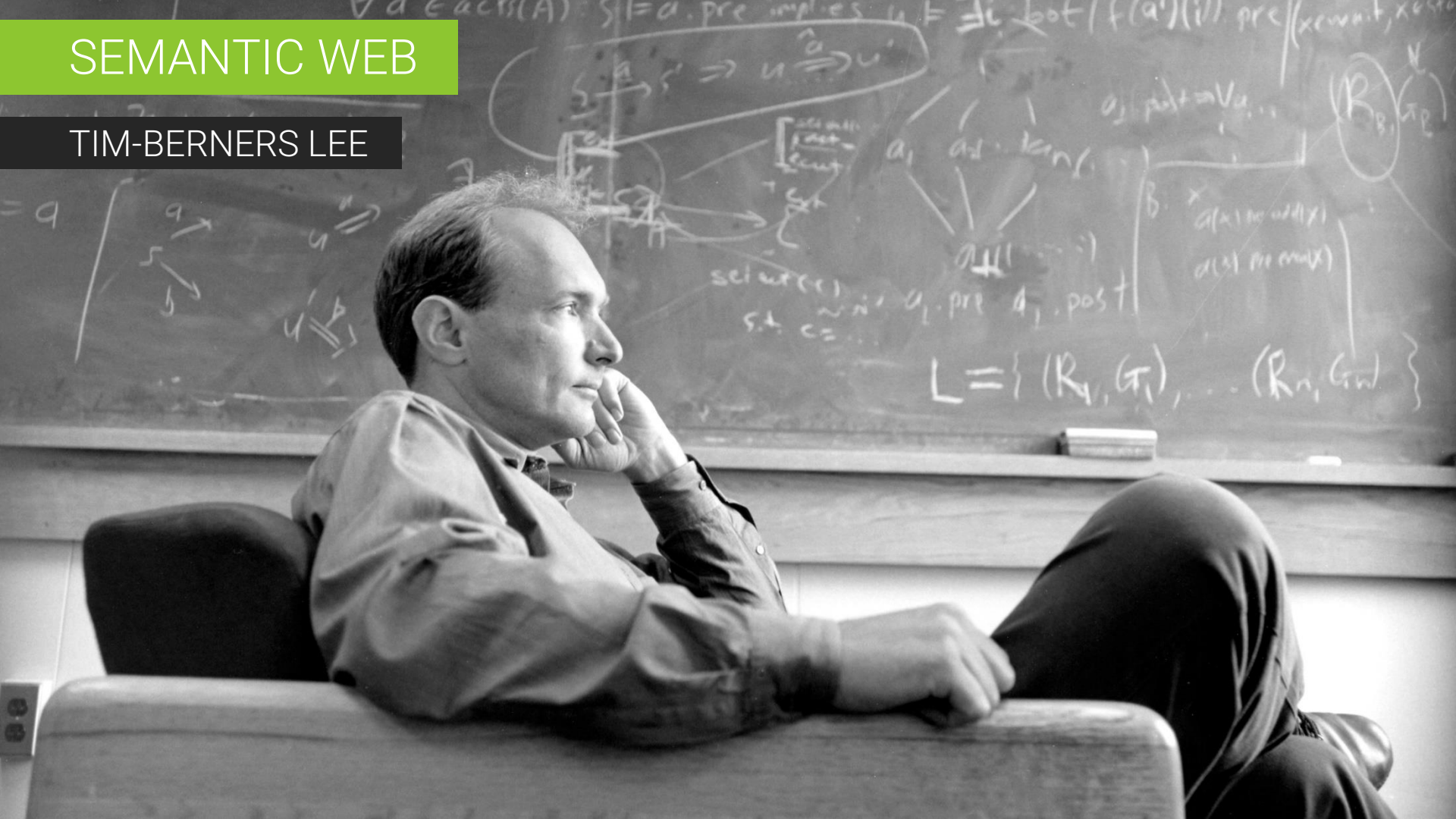
SEMANTIC WEB

THE LIMITATIONS OF THE CURRENT WEB

- If machines can actually understand the information in an unambiguous way, a smarter and more interconnected web becomes possible.
- Machines would be “first class” citizens of the web, just like humans.

SEMANTIC WEB

TIM-BERNERS LEE



SEMANTIC WEB

„The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.“

Tim Berners-Lee et al. 2001

- (1) It is an extension (an additional layer) of the current web
- (2) This layer defines the actual meaning (semantics) of the content in a machine-accessible way.
- (3) The goal is better human-machine cooperation.

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

INFORMATION

Declaration	**Universal data format** that can handle handle very diverse datasets and data-schemas, structured and unstructured data.
Storage	Both decentralized and centralized.
Retrieval	Standardized ways to retrieve data / information.
Aggregation	Ability to combine information from different sources.
Interpretation & Creation	Teaching machines the concepts behind the raw data. Enabling machines to create new information through logic.

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

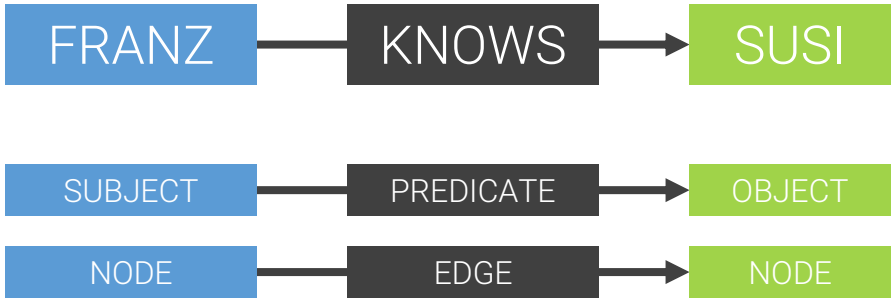
INFORMATION

Declaration	**Universal data format** that can handle handle very diverse datasets and data-schemas, structured and unstructured data.
Storage	
Retrieval	
Aggregation	
Interpretation & Creation	

SEMANTIC WEB

INFORMATION DECLARATION

- A simple, but very flexible data model called RDF is used.
- RDF is a data model concept, not an implementation!
- RDF's are Triples that form simple, grammatical statements:

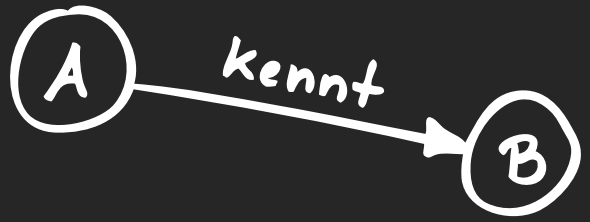


SEMANTIC WEB

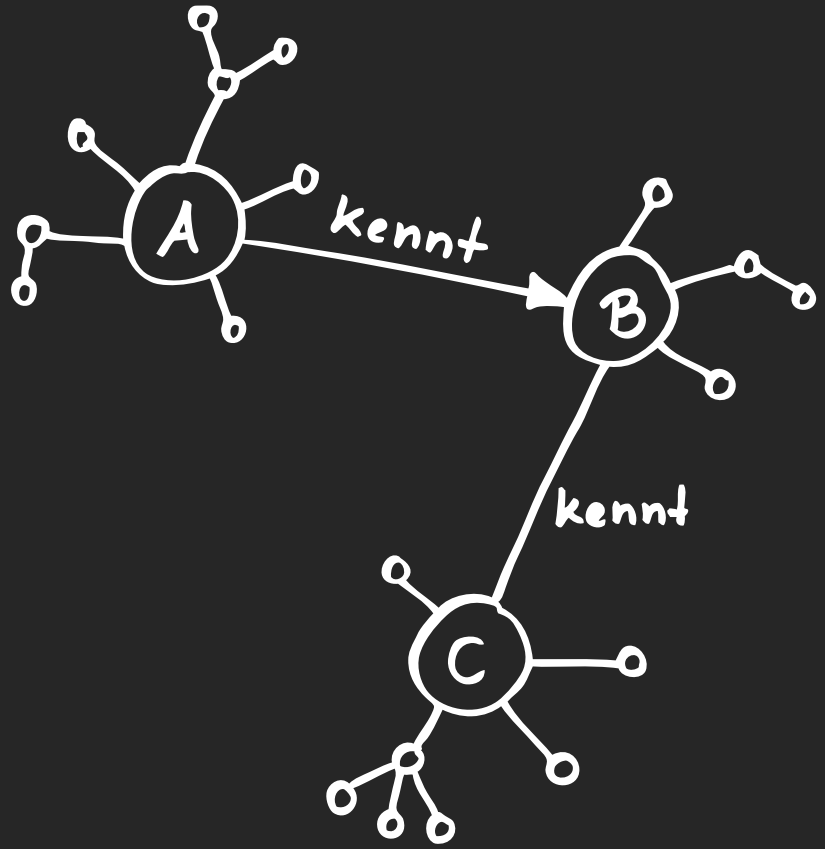
INFORMATION DECLARATION

- Each element of the Triple is an URI. This makes it unique through the whole internet

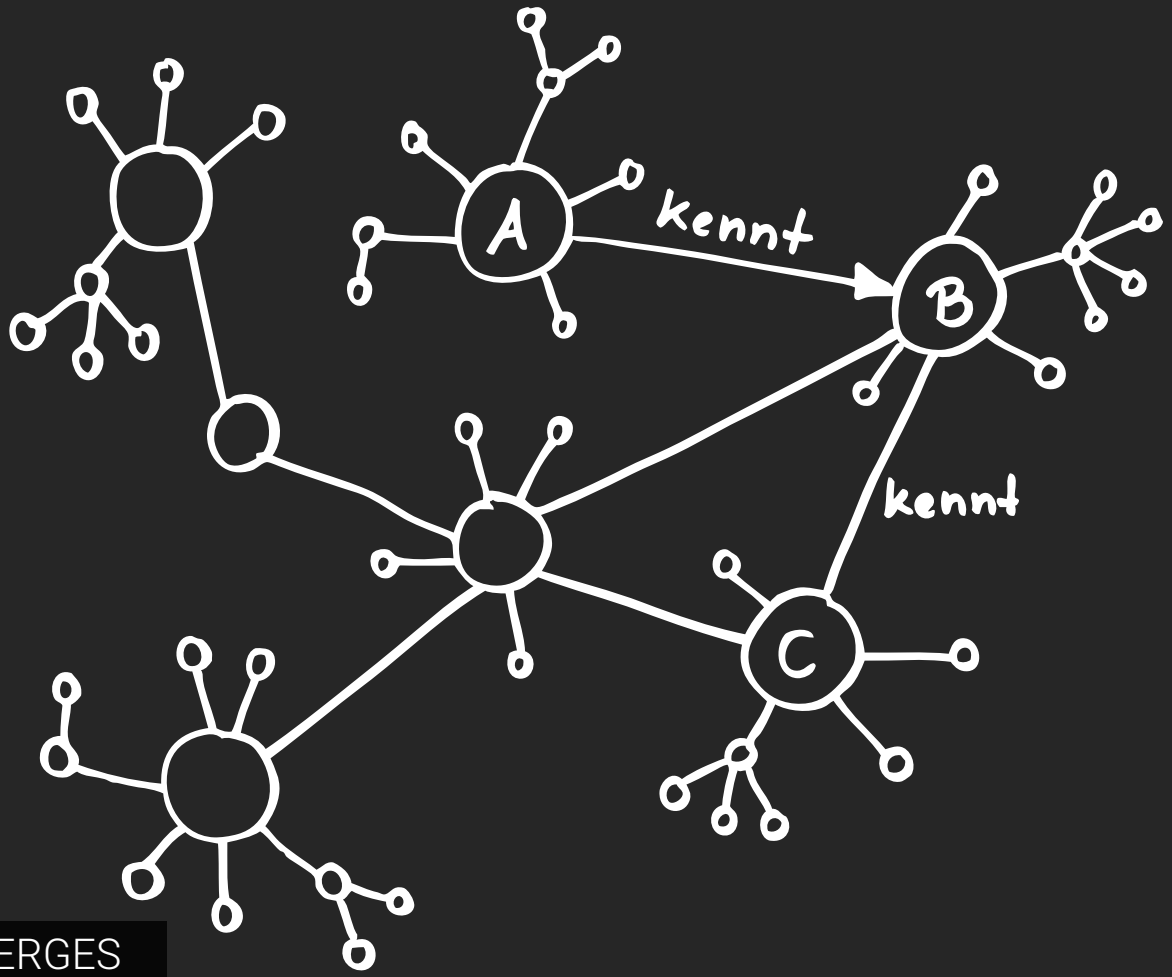
- URIs can also reference real-world or abstract things!
- If several statements share the same URI (talk about the same thing), they get linked together and form a graph structure.
- That’s why it’s called Linked Data.



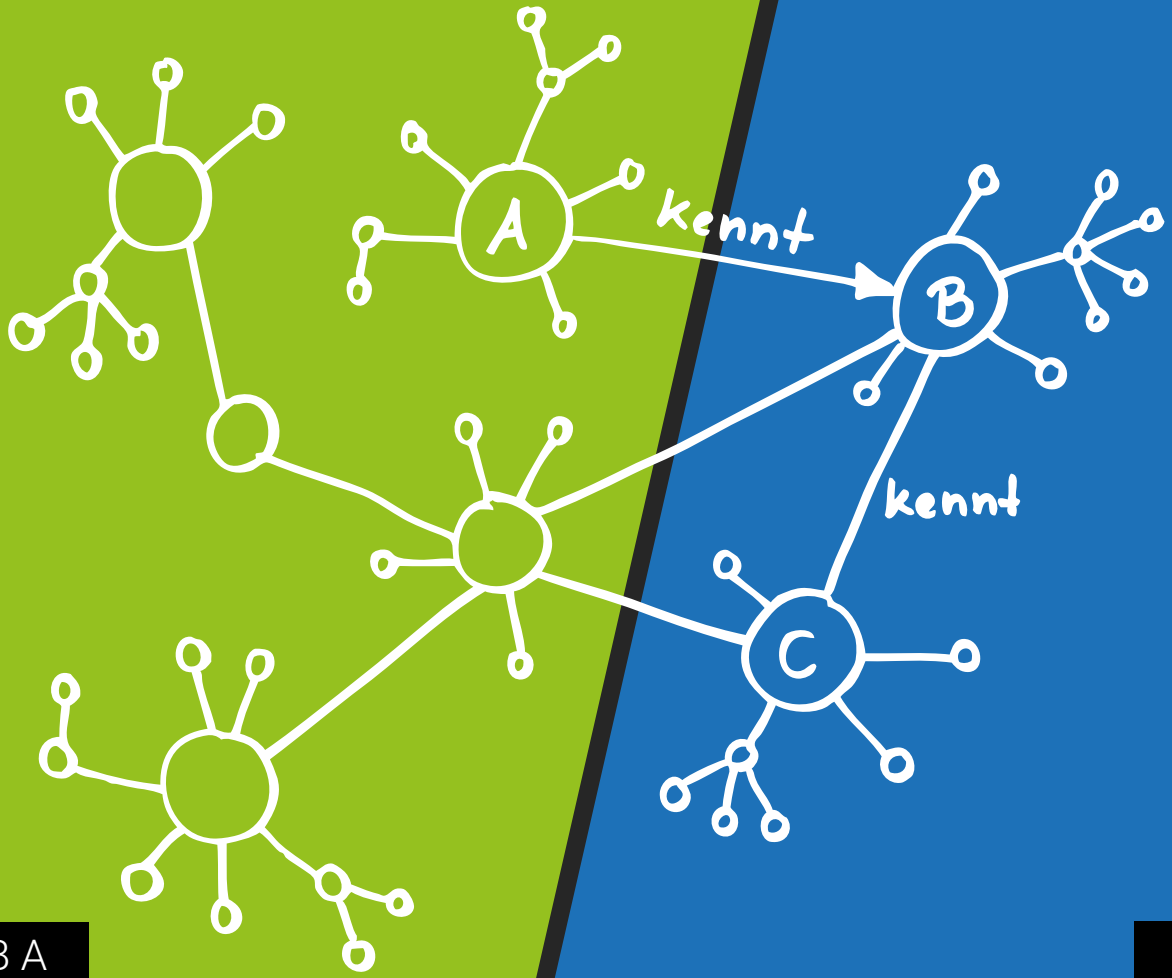
A „SINGLE“ TRIPLE



(A) AND (B) KNOW THE SAME PERSON (C)



A GRAPH EMERGES



WEBSITE / DB A

WEBSITE / DB B

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

INFORMATION

Declaration	
Storage	Both decentralized and centralized.
Retrieval	
Aggregation	
Interpretation & Creation	

INFORMATION STORAGE

- Decentralized storage is possible.
- RDF statements can be stored “behind” URLs.
  HTML with embedded Linked Data is a common use case.
- There are many different serialization formats for RDF.
  - XML/HTML based: RDFa, …
  - JSON based: JSON-LD, …
  - Plain-text based: Turtle, …

Without Markup | Microdata | **RDFa** | JSON-LD

```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Jane Doe</span>
  <img src="janedoe.jpg" property="image" />
  <span property="jobTitle">Professor</span>
  <div property="address"  typeof="PostalAddress">
    <span property="streetAddress">
      20341 Whitworth Institute
      405 N. Whitworth
    </span>
    <span property="addressLocality">Seattle</span>,
    <span property="addressRegion">WA</span>
    <span property="postalCode">98052</span>
  </div>
  <span property="telephone">(425) 123-4567</span>
  <a href="mailto:jane-doe@xyz.edu" property="email">
    jane-doe@xyz.edu</a>
  Jane's home page:
  <a href="http://www.janedoe.com" property="url">janedoe.com</a>
  Graduate students:
  <a href="http://www.xyz.edu/students/alicejones.html" property="colleague">
    Alice Jones</a>
  <a href="http://www.xyz.edu/students/bobsmith.html" property="colleague">
    Bob Smith</a>
</div>
```

Without Markup | Microdata | RDFa | **JSON-LD**

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Person",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Seattle",
    "addressRegion": "WA",
    "postalCode": "98052",
    "streetAddress": "20341 Whitworth Institute 405 N. Whitworth"
  },
  "colleague": [
    "http://www.xyz.edu/students/alicejones.html",
    "http://www.xyz.edu/students/bobsmith.html"
  ],
  "email": "mailto:jane-doe@xyz.edu",
  "image": "janedoe.jpg",
  "jobTitle": "Professor",
  "name": "Jane Doe",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}
</script>
```

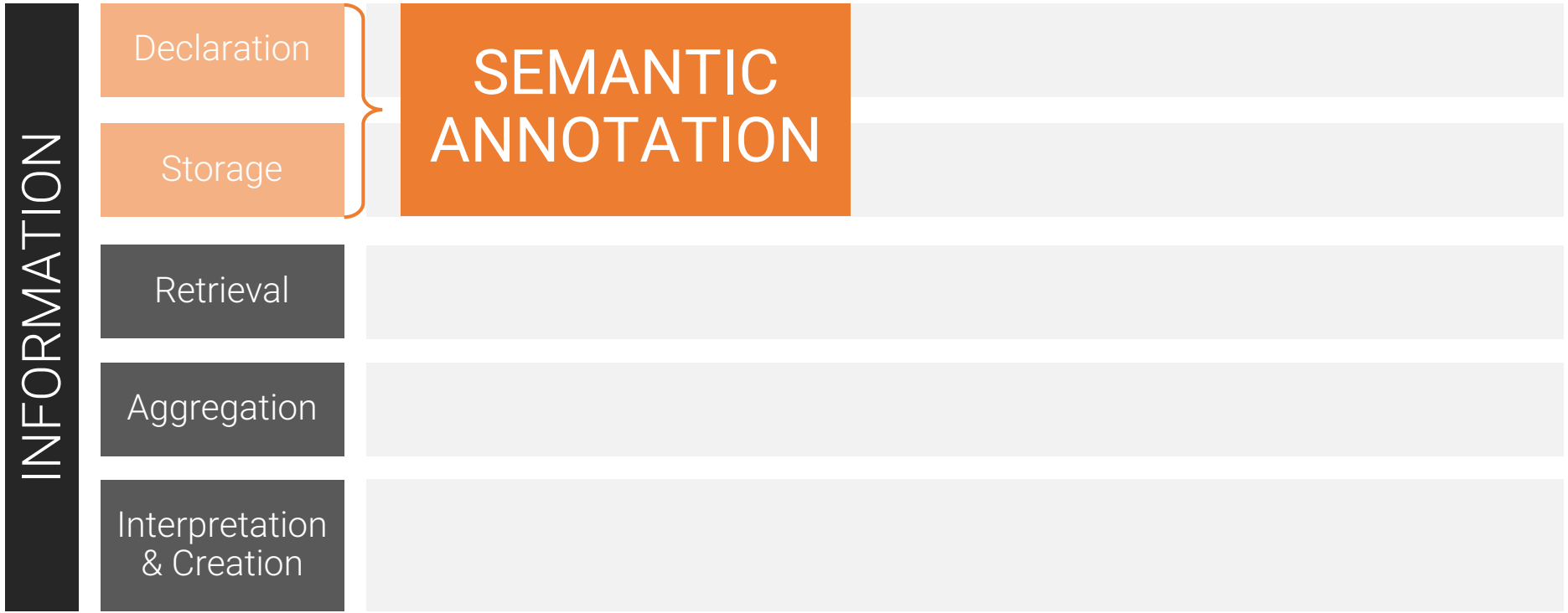
SEMANTIC WEB

INFORMATION STORAGE

- To store RDF in a central place, Triplestores are used.
- Triplestores are graph oriented databases that use RDF as native data model.
- They provide a standardized RESTful API to query, manipulate and access the data, called a SPARQL Endpoint.
- SPARQL is a standardized, graph oriented query language.

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB



SEMANTIC WEB

SEMANTIC ANNOTATION

- Semantic Annotation is the process of providing / adding machine-readable information on the web.
- It’s an advanced form of meta-data
- SEO is the discipline that comes closest, since its about machine-optimizing content.
- Example: schema.org

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

INFORMATION

Declaration	
Storage	
Retrieval	Standardized ways to retrieve data / information.
Aggregation	
Interpretation & Creation	

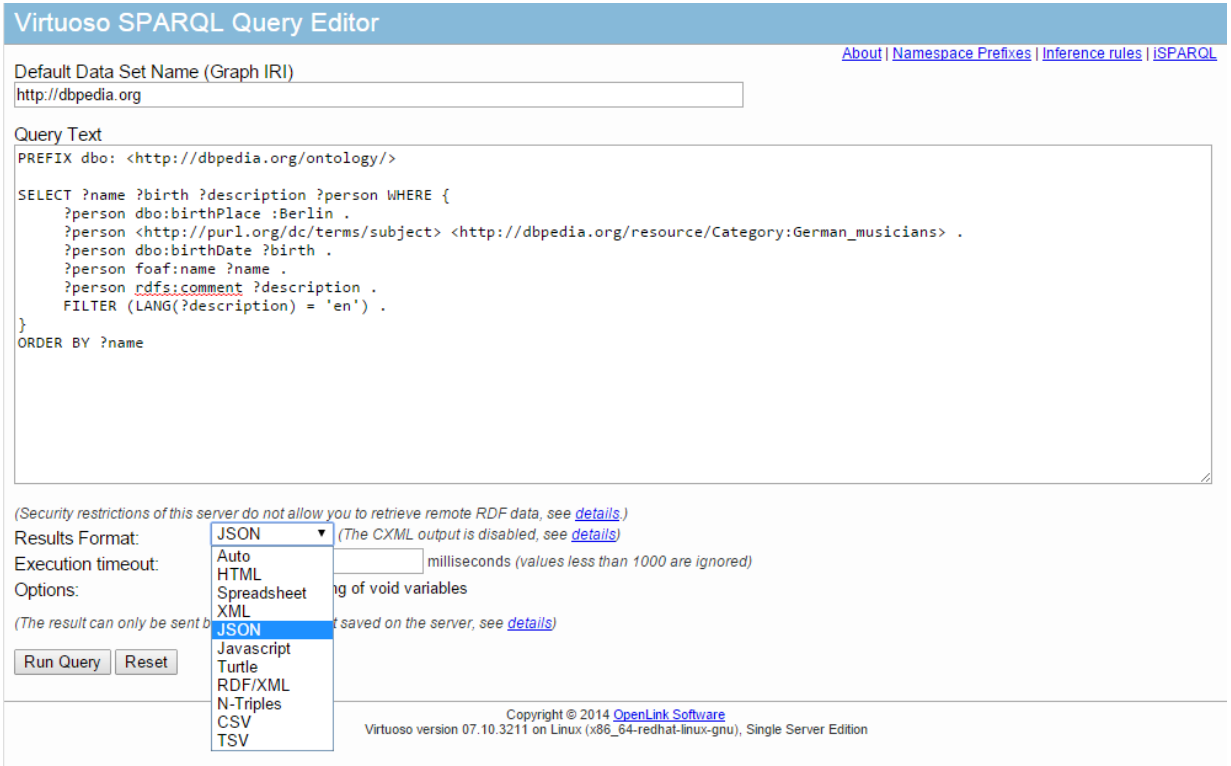
INFORMATION RETRIEVAL

- Decentralized:
  - Linked Data can be extracted by semantic crawlers.
  - Web browsers and Extension can add support, fetching the data from the current site and eventually crawling on.
- Centralized:
  - Linked Data can be queried at SPARQL endpoints.
  - Called “Linked Datasets”.

INFORMATION RETRIEVAL

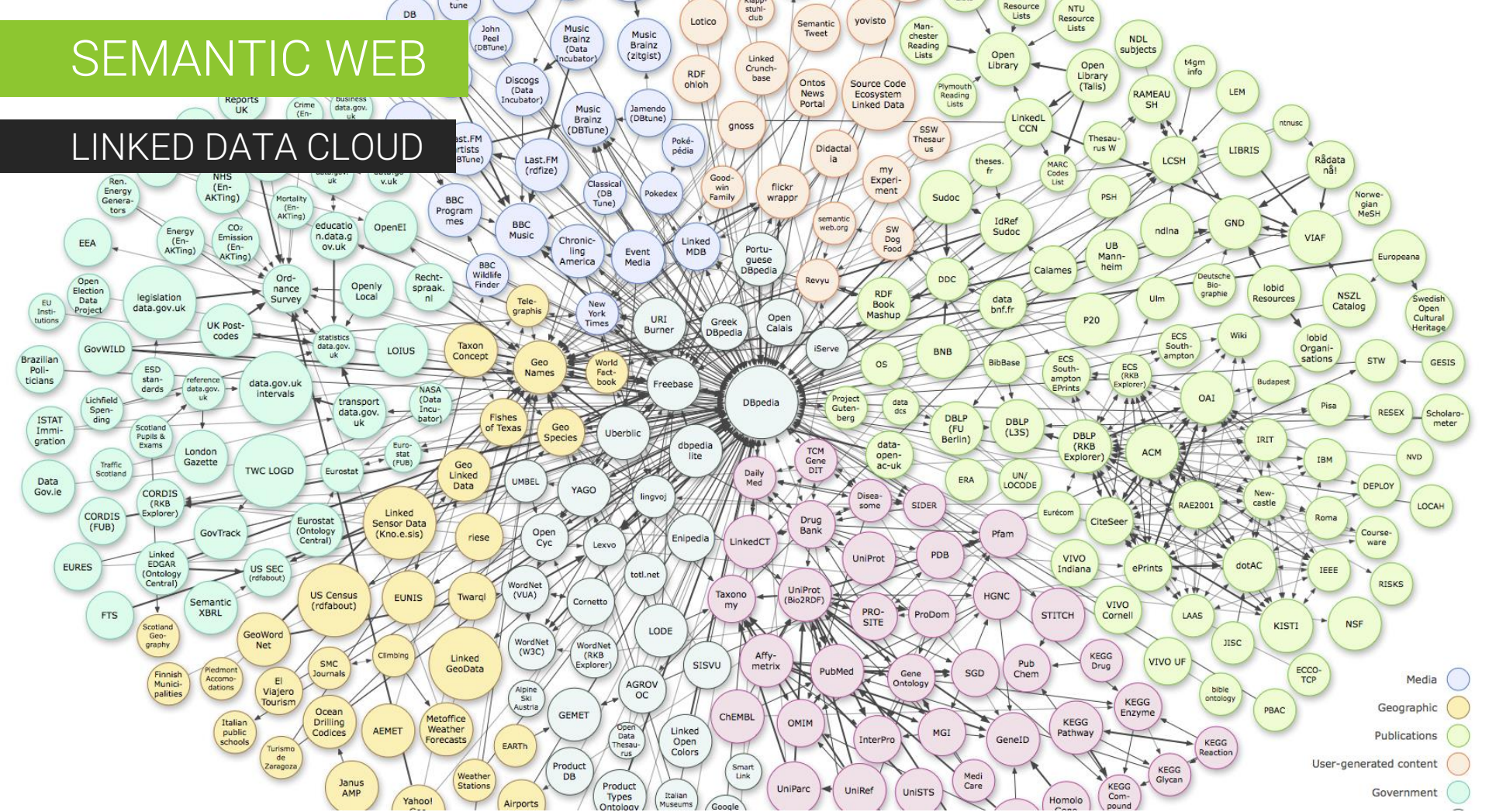
- Example: DBpedia extracts knowledge out of Wikipedia and makes it available as Linked Data.
- Allows for queries like “all german musicians born in berlin”

INFORMATION RETRIEVAL



SEMANTIC WEB

LINKED DATA CLOUD



SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

INFORMATION

Declaration	
Storage	
Retrieval	
Aggregation	Ability to combine information from different sources.
Interpretation & Creation	

SEMANTIC WEB

ABOUT GRAPH STRUCTURES

- A graph is the most flexible data model. Every other data structure can be described as a graph.
- Graphs can be easily merged, without compromising the data or its structure. (Trees or tables can’t be easily merged)
- Graph databases are schema-free: It is not required to define a schema before inserting data. The structure evolves naturally with/along the data.
  - Information can be saved “as they are”, they don’t need to be abstracted / reduced / adjusted before inserting.

INFORMATION AGGREGATION

- Since RDF is graph oriented, merging is easy
- Worst case: No common elements or relations resulting in a unconnected graph.
- SPARQL supports querying and merging data from different databases on the fly!

SEMANTIC WEB

CHALLENGES OF THE SEMANTIC WEB

INFORMATION

Declaration	
Storage	
Retrieval	
Aggregation	
Interpretation & Creation	Teaching machines the concepts behind the raw data. Enabling machines to create new information through logic.

SEMANTIC WEB

INFORMATION CREATION

- Graphs are schema-free but it is possible to add a schema on top.
- Data and schema stay separated and independent. It is possible to change the schema without migration issues.
- Semantic Web uses Ontologies, which are very powerful and sophisticated. They are based on first-order-logic.

SEMANTIC WEB

INFORMATION CREATION

- Ontologies describe how things relate to each other.
- Analogy to natural language: It is not sufficient to have the same grammar (syntax, data formats) and vocabulary (words, URIs). A common understanding of a domain is required for successful communication.
- Ontologies allow for reasoning and inference.
  New knowledge can be deducted through logic.
- Think of: Google Now, Apple Siri, IBM Watson

CONCLUSION

- SEO might be the main driver for adoption of the Semantic Web, since it provides immediate and practical value as soon as the big companies decide to reward it.
- SEO only requires know-how about Semantic Annotation, which is a rather easy part of the SW Technologies.

SEMANTIC WEB

CONCLUSION

- The Semantic Web is a big vision, maybe on par with “world peace”. Many people would say it’s not realistic.
- Maybe that’s true, but even trying could bring great value and improvement to the current situation.
- It’s a “top down” development. That makes its adoption uncertain.

Roles of
The Future Web

Web as a Platform

WEB AS A PLATFORM

THE UBIQUITOUS WEB

- There is almost no OS that doesn’t support the web as application platform.
- Trend toward OSs that use the Web as its main, native platform: ChromeOS, WebOS, FirefoxOS.
- Many new types of web-enabled devices are appearing: SmartTV’s, SmartWatches, SmartCars and even SmartFridges.

WEB AS A PLATFORM

THE UBIQUITOUS WEB

- With Node.js JavaScript runs outside of browsers, providing an interpreter and package manager just like Python or Ruby.
- JavaScript can run on both server and client, allowing them to share the same code. (Isomorphic JavaScript)

WEB AS A PLATFORM

JAVASCRIPT AS A COMPILE TARGET

- JavaScript is increasingly used as compile target from different languages.
- There are projects like ASM.js that provide additional boosts in performance by providing optimization annotations.
- WebGL and WebCL enables the usage of GPU computing
- WebWorker enable multi-threading.
- JavaScript usually runs in a sandbox, which is a big security bonus.

WEB AS A PLATFORM

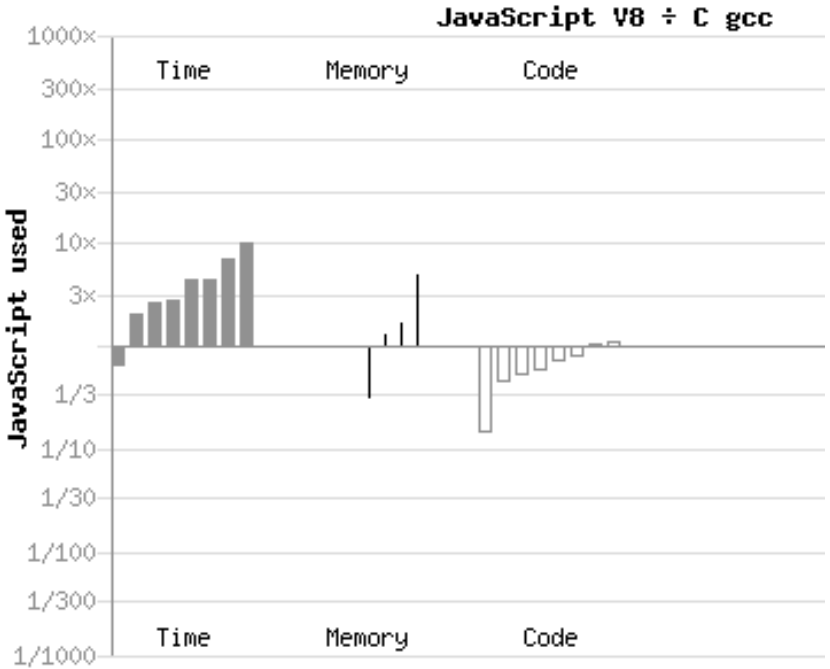
JAVASCRIPT PERFORMANCE

- Huge performance optimizations went into JavaScript since Google released the V8 engine in 2008.
- How fast is JavaScript actually?
- Performance is not a nice-to-have, it is a feature!
- Modern Web Apps would not have happened with slow JavaScript engines.
- WARNING: Benchmarks are like statistics… you can prove anything. Usually they are very artificial.

WEB AS A PLATFORM

http://benchmarksgame.alioth.debian.org/u32/benchmark.php

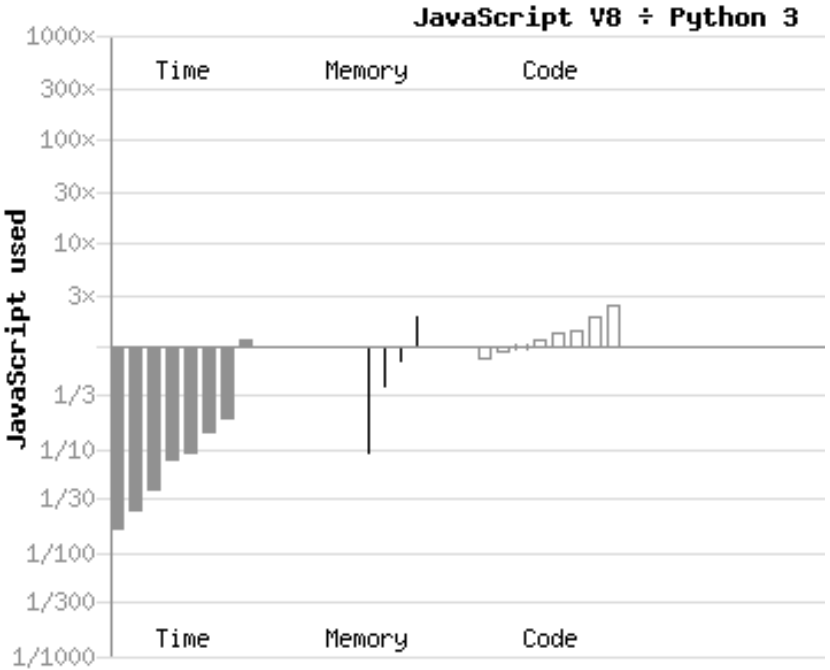
JAVASCRIPT <-> C



LOGARITHMIC SCALE!

~2-10 times slower than C

JAVASCRIPT <-> PYTHON

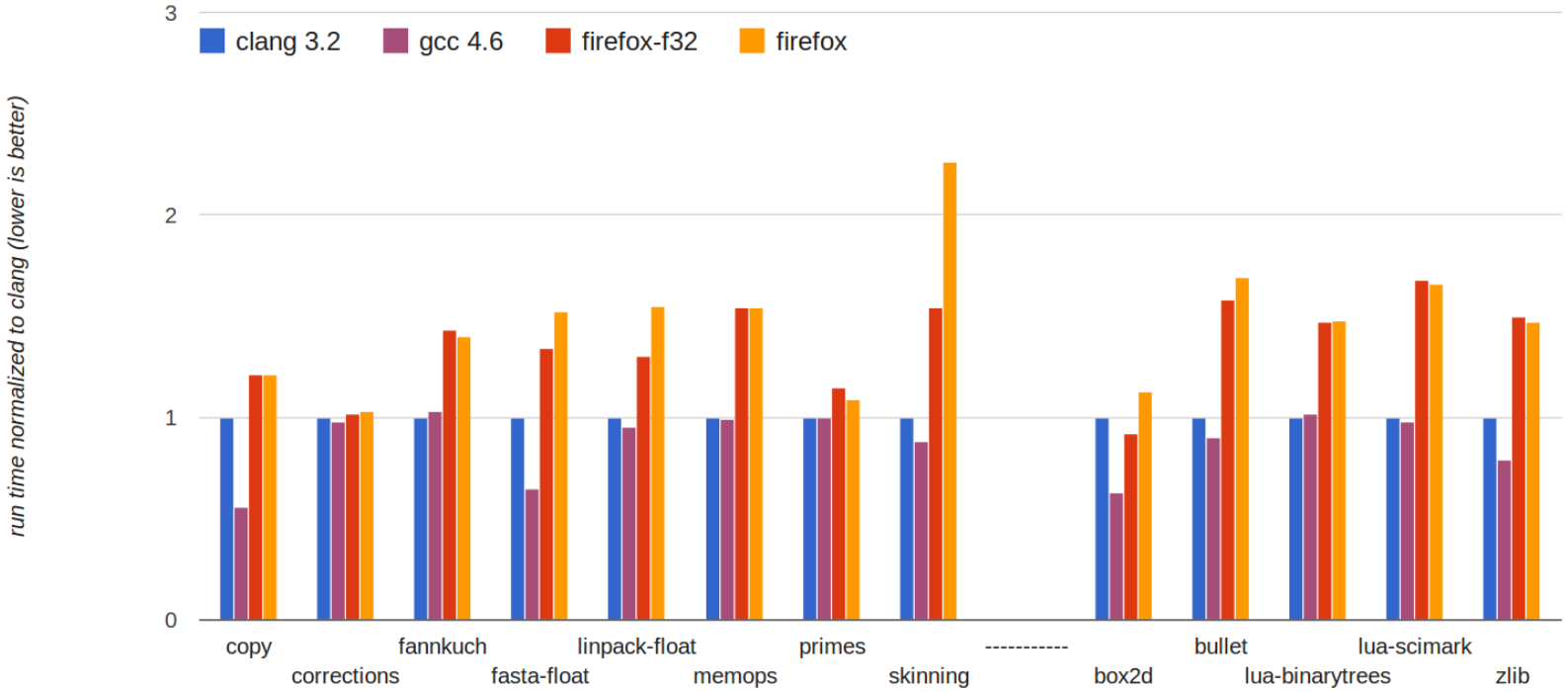


~5-50 times faster than Python

WEB AS A PLATFORM

https://hacks.mozilla.org/2013/12/
gap-between-asm-js-and-native-performance-gets-even-narrower-with-float32-optimizations/

ASM.js <-> NATIVE C



WEB AS A PLATFORM

JAVASCRIPT AS A COMPILE TARGET

- Even big and complex applications can be ported:
  - Linux running in the browser: www.bellard.org/jslinux
  - Unreal Engine 4: www.unrealengine.com/html5
- A programming language of choice can be used to write web applications.
  - C++ (emscripten)
  - JAVA (GWT)
  - Dart, CoffeScript, TypeScript
  - New languages appearing every few months…

WEB AS A PLATFORM

CONCLUSION

- The web is the only platform that runs nearly everywhere (ubiquitous).
- This is even more true for JavaScript, which is becoming the default cross-platform “virtual machine bytecode”. JavaScript succeeded where JAVA failed.

Web of Applications

WEB OF APPLICATIONS

CURRENT STATE

- Most of the web is still a “Web of Documents”.
- Current CMS are almost exclusively built the traditional stateless way, serving sites as documents.
- This is very inefficient and slow, since the whole site and all resources have to be (re)calculated and (re)loaded after every click.
- White flash effect!

WEB OF APPLICATIONS

OUTLOOK

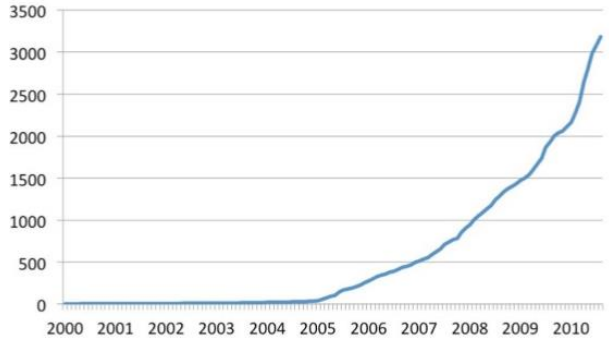
- Web Apps are often Single-Page-Applications (SPA). They load only once and conditionally load new data through APIs.
- This was bad fore SEO, but Google started adding support for indexing “AJAX Applications”.
- Maybe future websites will feel & behave like web applications?
- Servers would be mainly providing API’s and persistance.
- Users would get used to near-instant feedback. (< 100ms)

Web of Services

WEB OF SERVICES

WEB APIS

- The web consists not only of the visible part of websites.
- There is a big, accelerating trend to (RESTful) Web-APIs.
- The web is increasingly used by machines.



Total APIs over time

http://de.slideshare.net/jmusser/j-musser-semtechjun2011

WEB OF SERVICES

INTERNET OF THINGS

- With the advent of the Internet of Things, Services become even more important.
- The IoT and the Web share the same platform: The Internet.
- The IoT could contribute new ideas and technologies to the Web of Services and vice versa.
- Examples: MQTT and WebSockets

Résumé

RESUME

- Web Development is currently improved and modernized, but developers have often to wait for several years to make use of it.
- That’s the price we pay for a open and decentralized web.

RESUME

- We’re heading toward an ubiquitous web, becoming the most widespread and important software-platform.
- The Web is not only happening inside the browser window.
  - It is enters many new devices and roles.
  - The invisible Web of Services and Web of Data is growing fast in size and importance.
- Machines play an increasing role in the new web, providing smarter and more interconnected applications.
- The future web is a powerful basis for Human-Machine cooperation.

Questions?

Thanks for listening!

www.fannon.de/url/swp-ppt