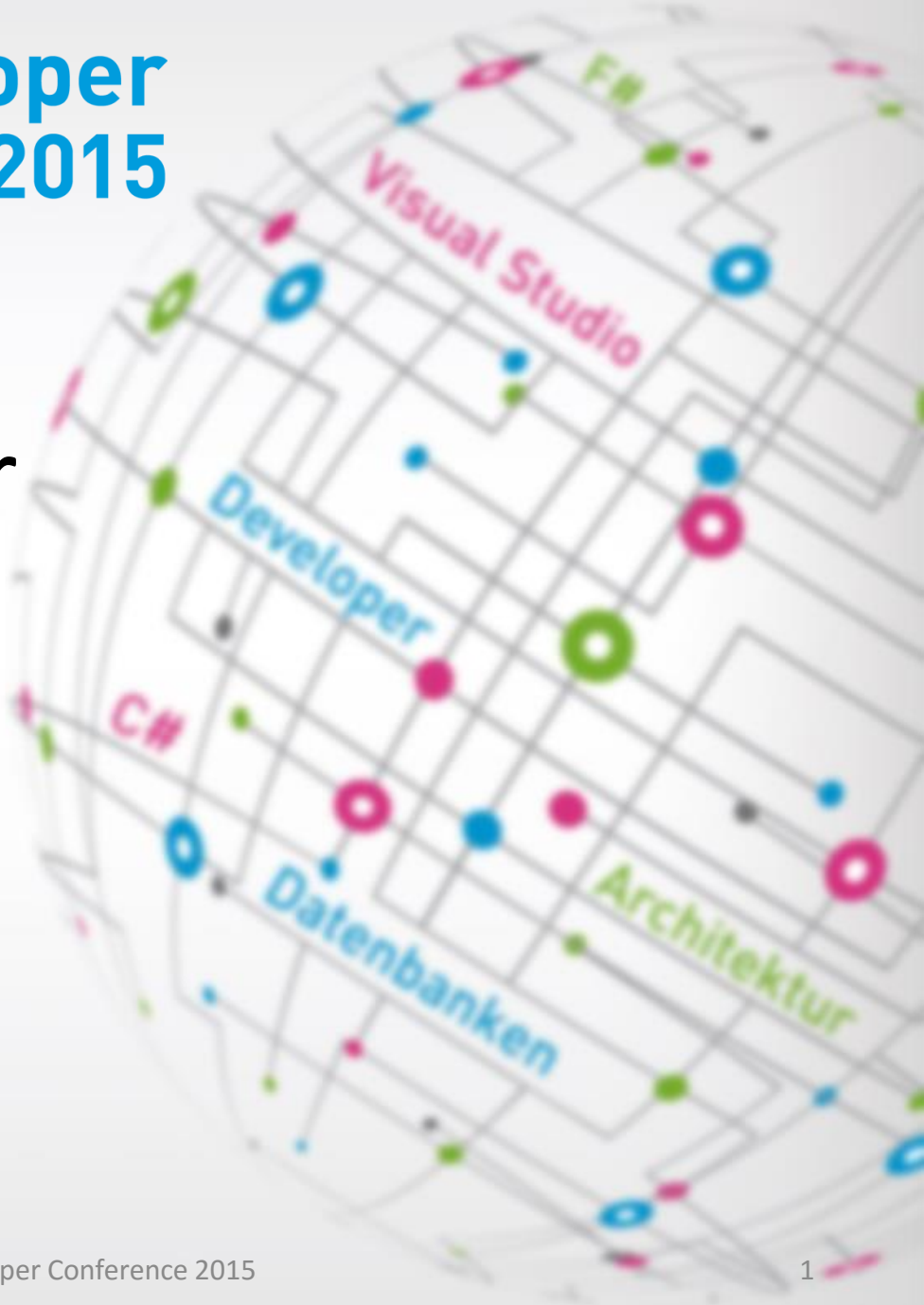
.NET Developer Conference 2015

#netdc15

Der Traum der perfekten API

Florian Rappl



Architecture and Design

IS THERE A PERFECT API?

Florian Rappl

#netdc15
@ddc_conference

- IT consultant
- Theoretical particle physics
- Microsoft MVP
- CodeProject MVP
- Library author and contributor

What is an API?

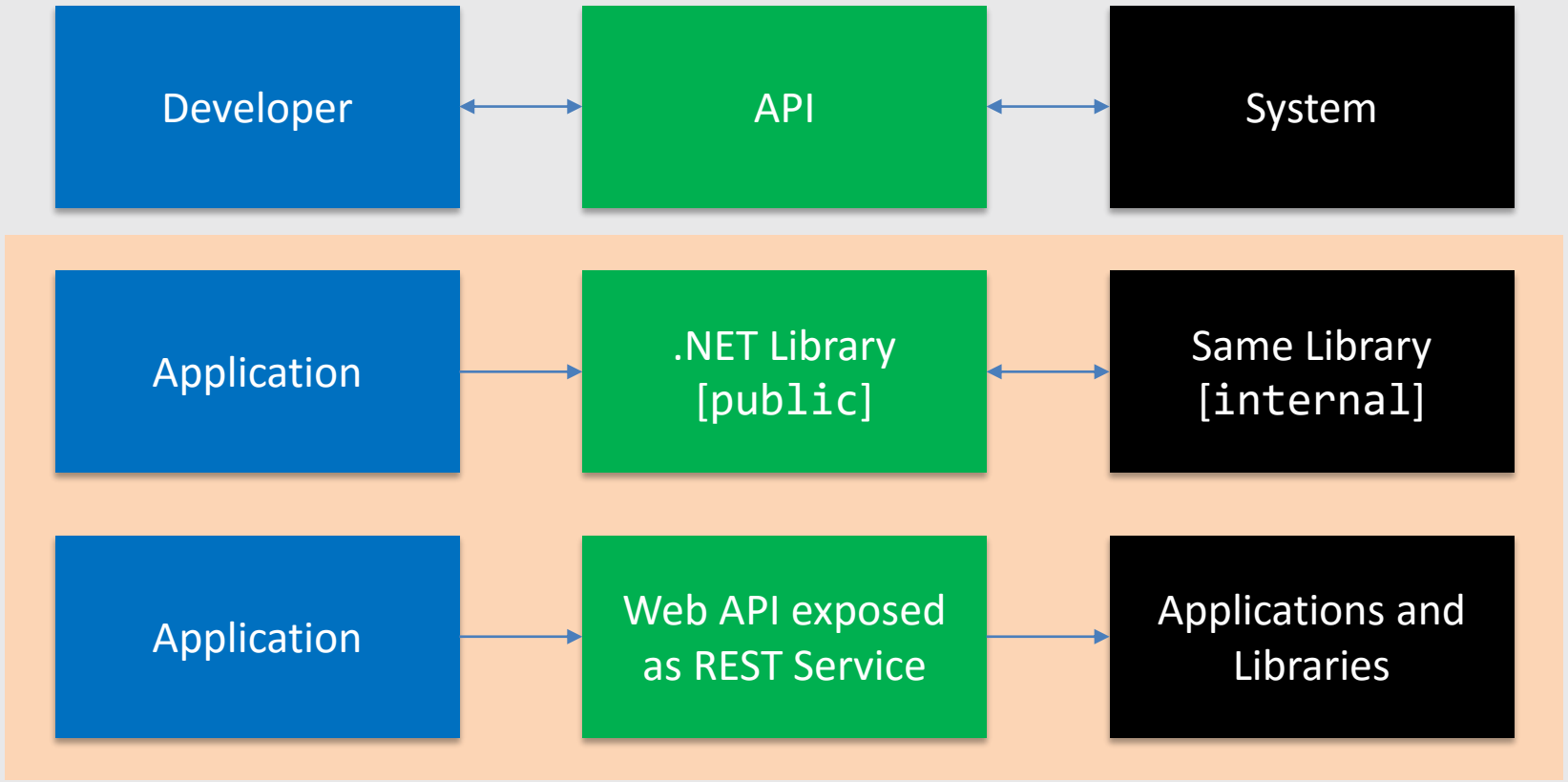
#netdc15
@ddc_conference

From Wikipedia:

„An application programming interface (API) is a set of routines, protocols, and tools for building software applications. […] An API defines functionalities that are **independent** of their respective implementations, which allows definitions and implementations to **vary without compromising** the interface.“

What is an API?

#netdc15
@ddc_conference



Reasons

#netdc15
@ddc_conference

- Why do we need good API design?
  - Reduce cost of development
  - Improve API adoption
  - Separation of concerns

A good API ...

#netdc15
@ddc_conference

- … is easy to understand
- … feels familiar, yet fresh
- … is simple to extend and adjust
- … avoids being misused
- … tends to be unobtrusive

#netdc15
@ddc_conference

Powerful, yet dead simple.

Easy to understand

#netdc15
@ddc_conference

- Prefers domain language
- Is consistent / transitive
- Documented
- Avoids side-effects

Do **not** invent words!

`Push()` => `Pop()`

```
///<summary>
```

`CheckUser()` should **not** start a session!

Familiar, yet fresh

#netdc15
@ddc_conference

- Follows
  - language conventions — Potential case-problems
  - framework design — `ITransport`
  - best practices — Avoid `out` parameters
- Makes use of patterns — Dependency Injection
- Reuses framework elements — `Uri`, `DateTime`, …

Extensible

#netdc15
@ddc_conference

- No artificial constraints
- General input (parameters)
- Most specific output (return value)
- Allow customization
- Not limited to very specific scenarios

➔ FCoI and SOLID

Avoids misuse

#netdc15
@ddc_conference

- Guides the user `Serial.Check(ProcessorId processorId)`
- Clean Code Good, clear names
- Throws as soon as possible

```
if (arg == null)
{
        throw new ArgumentNullException(nameof(arg));
}
```

Unobtrusive

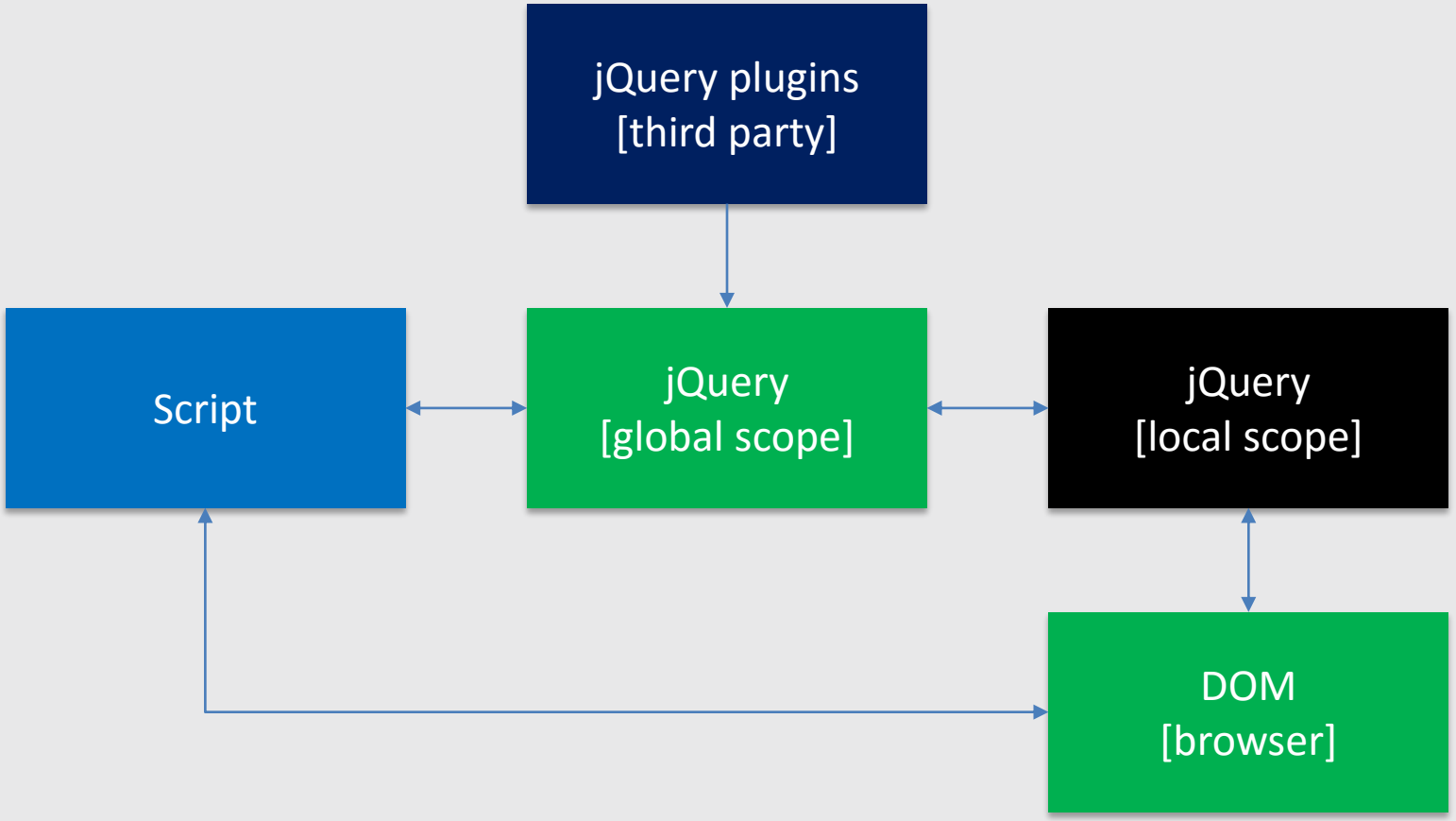
#netdc15
@ddc_conference

- No (fixed) dependencies — Be agile!
- Minimum configuration — Coding by convention
- Adjustable resources — Don‘t start your own thread pool!
- Easy to integrate / desintegrate
  - Mainly expose interfaces
  - Prefer framework elements
  - Allow custom types

Some case studies.

jQuery

#netdc15
@ddc_conference



jQuery

#netdc15
@ddc_conference

```
$("#button-container button").on("click", function (
event) {
  $("button.continue").html("Next Step...");
});

$.ajax({
  url: "/api/getWeather",
  data: { zipcode: 97201 },
  success: function (data) {
    $("#weather-temp").text(data + " degrees");
  }
});
```

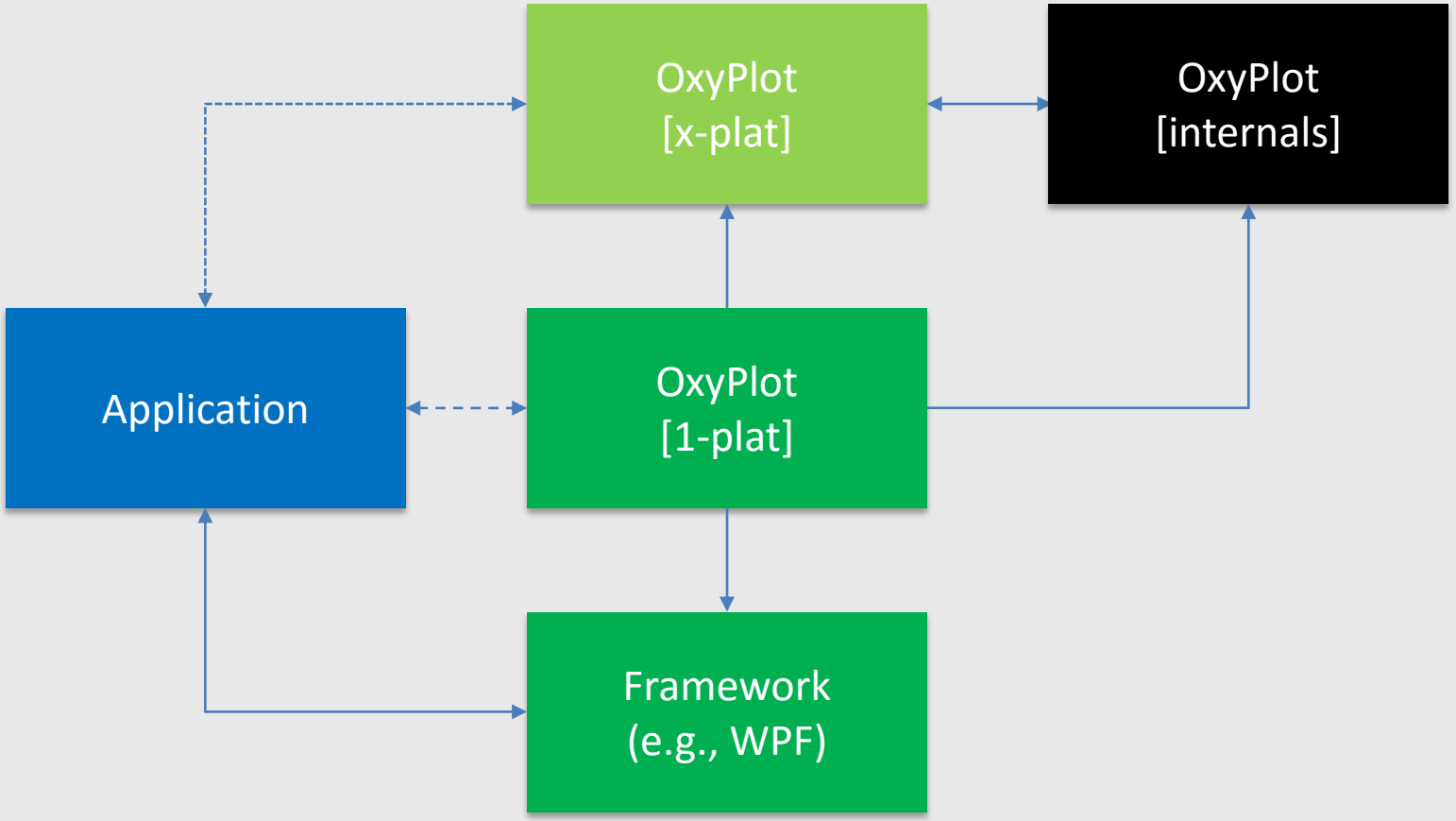
jQuery

#netdc15
@ddc_conference

- (Abstraction) Layer over the DOM
- Fluent API
- Very easy to extend
- Sometimes multiple ways to do things
- Can be misused (perf. hits) easily

OxyPlot

#netdc15
@ddc_conference



OxyPlot

#netdc15
@ddc_conference

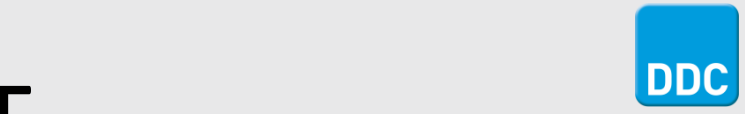
```
var model = new PlotModel { Title = "Simple example" };

var series1 = new LineSeries { Title = "Series 1",
MarkerType = MarkerType.Circle };
series1.Points.Add(new DataPoint(0, 0));
series1.Points.Add(new DataPoint(40, 15));

var series2 = new LineSeries { Title = "Series 2",
MarkerType = MarkerType.Square };
series2.Points.Add(new DataPoint(0, 4));
series2.Points.Add(new DataPoint(40, 5));

model.Series.Add(series1);
model.Series.Add(series2);
```

OxyPlot

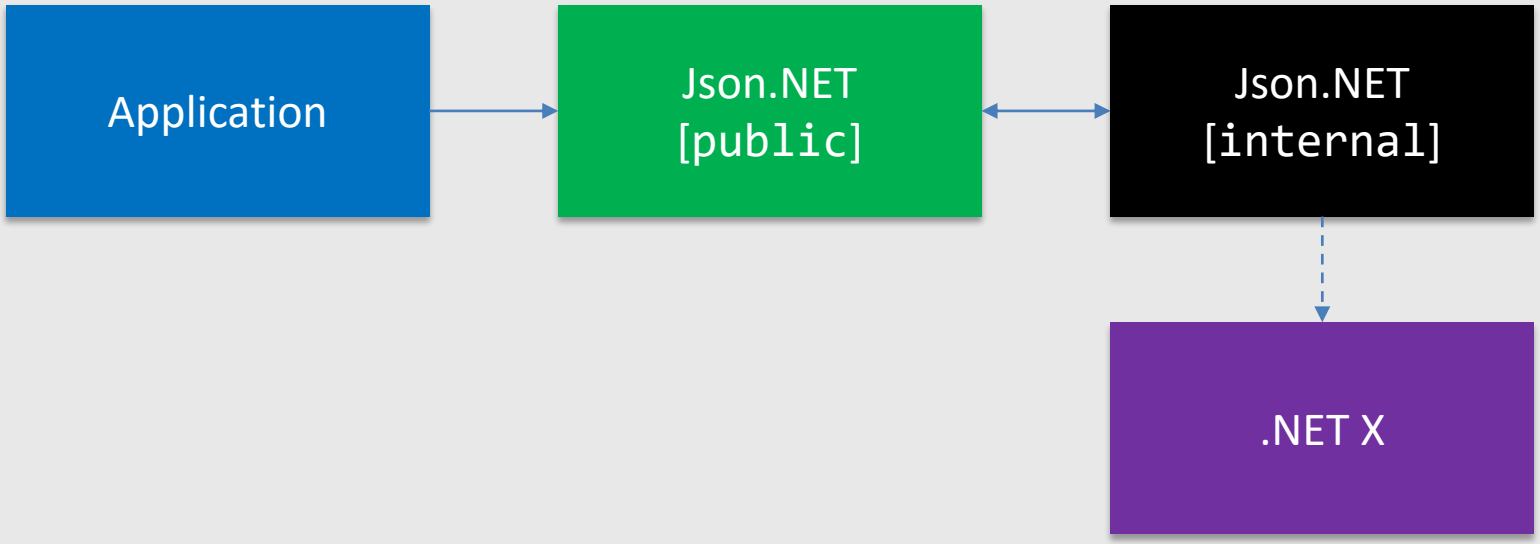
#netdc15
@ddc_conference

- .NET x-plat plotting library
- Easy to extend
- Simple for trivial scenarios
- Hard for complicated scenarios
- Misuse is possible, but not easy

Json.NET



#netdc15
@ddc_conference



Json.NET

#netdc15
@ddc_conference

```
Product product = new Product();
product.Name = "Apple";
product.Expiry = new DateTime(2008, 12, 28);
product.Sizes = new string[] { "Small" };
string json = JsonConvert.SerializeObject(product);

Movie m = JsonConvert.DeserializeObject<Movie>(json);
```

Json.NET

#netdc15
@ddc_conference

- .NET JSON serializer / deserializer
- Balanced extensibility
- Many side-effects
- Simple for most scenarios
- Nearly impossible to misuse

Design Guidelines

#netdc15
@ddc_conference

- Gather requirements
- SPI – Write at least 3 plugins before release
- Expect to evolve API
- Maximize Information Hiding
- Document religiously
- Minimize mutability

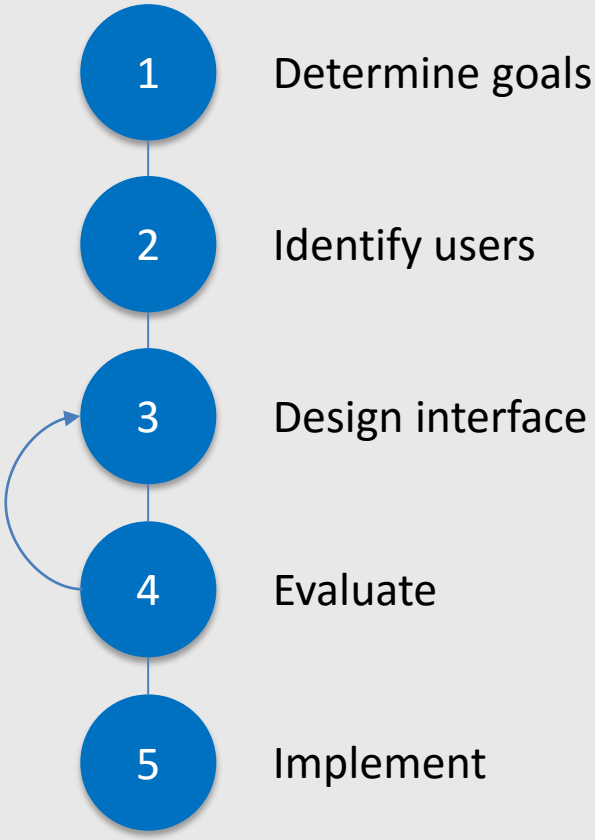
Common Trends

#netdc15
@ddc_conference

1. Provide “just enough” functionality
2. User-centered design approach
3. Do not break existing applications with API changes

Iterative & User-Centered

#netdc15
@ddc_conference



#netdc15
@ddc_conference

What can we learn?

Lessons

#netdc15
@ddc_conference

- Simple things need to be really simple
- Coherent design decisions
- We need at least minimal extensibility
- Limit possible ways to guide users
- One scenario, unlimited possibilities

Conclusions

#netdc15
@ddc_conference

- There is not one perfect API
- We know the basic recipe for a good API
- Overall design is mostly aesthetics
- Most importantly:
  - Mix technological superior design with creativity
  - Be consistent, always

#netdc15
@ddc_conference

Don‘t be afraid to iterate until you find the perfect API

(but do so by using *semver*)

@florianrappl

florian-rappl.de

mail@florian-rappl.de

- **Downloads** unter bit.do/netdc15download
- **Feedback**: kerstin.hartmann@developer-media.de
- **Follow** us:
  fb.com/DDConference
  @ddc_conference
- Wir sehen uns:

SMART DATA Developer Conference
Big Data Analytics: 18.-19.April, München

Developer Week 20.-23.Juni, Nürnberg

DDC 2016 05.-07.Dezember, Köln