We Are Developers!

Summer 2/2022

A Special Issue by
Heise Medien GmbH & Co. KG

> WEB DEVELOPMENT

A New Generation of Web Frameworks

> THE STATE OF JAVA

Interview: Nine Experts on Java 17 and Beyond

> DIVERSITY IN DEVOPS

How to Increase Gender Diversity

> TYPESCRIPT

Tutorial: Low Maintenance Types

> CLIMATE CHANGE

Building Climate-Friendly Websites

> MACHINE LEARNING

Adversarial Attacks and Defenses

Powered by heise Developer

EDITORIAL

Welcome to the Future

Dear reader,

the future is often on our minds, as numerous popular works of fiction have been proving for centuries. They may include time travel, such as the first literary description of time travel in H. G. Wells' social critique "The Time Machine," published in 1895, or be set altogether in a distant future. Over time, the idea of taking a glimpse at scenarios the future may hold does not seem to have lost any of its charm. Time travel and visions of the future are often found in contemporary TV series, movies, and books. The depictions range from a dystopian view of cybernetic entities ruling over humanity to a glorifying outlook, such as a peaceful planet Earth, which is populated by humans who no longer know war or hunger and who share the same goal of exploring the universe.

Thinking of the less distant and more realistic future, there are several uncertainties young developers face when entering the job market. You may ask yourself whether the technology you have specialized in is up to date and desired by prospective employers, how to improve your skills, or how to find a company where you will feel comfortable working. This magazine tries to provide some answers. The programming language TypeScript, for example, is a superset of JavaScript that offers additional options to make developers' lives easier. Among these are low maintenance types, which Stefan Baumgartner describes in a detailed tutorial. While current web frameworks are reaching their limits, a new generation of frameworks is on the horizon. Miško Hevery explains why these must be "replayable and resumable." A look into the future is also what nine Java experts attempted in a comprehensive interview about the current and (possible) future state of Java. Stefania Chaplin dives into her own experiences of working at companies that value diversity. She provides insights into how companies can increase gender diversity in their DevOps teams, changing them for the better—after all, a diverse team can bring new ideas and perspectives to the table. Christoph Reinders and Bodo Rosenhahn take a look at possible adversarial attacks on neural networks and how this may affect driver-assistance systems in autonomous vehicles—bringing up not dystopian, but very real concerns about potentially fatal errors. And since there can only be a future for us if climate change is slowed significantly, Roy Derks stresses the importance of the role developers play, because how websites are built can have a large impact on conserving energy.

Live long, prosper, and enjoy reading the articles in this magazine,

Maika Möbus

CONTENTS

Young Professionals Write for Young Professionals

This journal is based on a series of articles published by Heise Online, where we provide a platform for young professionals to publish their first professional articles. The journal is released in German language in a six-monthly rhythm, and is now arriving in English for the first time. For writing, the authors receive mentoring from the Heise Developer editorial team. The series is also intended to encourage young people to develop their talent as authors; for example, to share important experiences with your peers or to present an own project you are engaged in. Or simply because you have always wanted to write a technical article, preferably in German.

Your First Professional Article for Heise

Contact: developer@heise.de

> Low Maintenance Types in TypeScript

Stefan Baumgartner

Dependent types are an interesting feature in TypeScript: Not only do they show what is happening in a program at any given time, but they also prepare the code for things to come.

The ultimate goal of TypeScript is to formalize even the most versatile interfaces in JavaScript, an inherently flexible and dynamic programming language. TypeScript is a superset of JavaScript with a syntax that enables static typing. It adds a variety of operators and constructs to the language to describe those complex scenarios. But not only do they allow for basic formalism, they also let developers create dependencies between types. Those typing constructs can be used to create a set of low maintenance types: static types, which are dynamic enough to update themselves if certain conditions change.

TypeScript's Type System

TypeScript's type system has one simple rule: A type defines a set of compatible values. As an example, the number 1337.42 is a compatible value of the type `number`. The string "Hello, World!" is compatible with the type `string`. Both `string` and `number` are primitive types and define a theoretically endless set of values. The primitive type `boolean`, on the other hand, only defines two values: `true` and `false`.

In a Nutshell

- > TypeScript is a superset of JavaScript that adds operators and constructs for static typing.
- > Its syntax allows for creating dependencies between types.
- > Developers can use this to create low maintenance types that can update themselves.

Interfaces and type aliases allow the description of complex JavaScript objects. These are called compound types, as they are usually an assembly of various properties of primitive types. The definition of a compound type also enables defining a set of compatible object values. For example, the type `Person`

```
type Person = {
  age: number,
  name: string
}
```

describes all objects which contain a property `age` of the type `number`, and a property `name` of the type `string`. Objects with additional properties are also compatible with the type `Person`. The shape of object types is important. Object types are compatible with each other if the same properties have the same types (Listing 1).

Compared to other programming languages which—with a few notable exceptions—feature nominal type systems, TypeScript features a structural one: As long as the shape of an object is equal or similar, TypeScript considers this a valid type check. A nominal type system, on the other hand, would not only ensure the shape to be the same, but also the assigned name to be the same or to be in a compatible hierarchy.

The reason for using a structural type system instead of a nominal one lies in the way JavaScript works: Object literals are everywhere, so a nominal type system would only prevent developers from using what already exists.

TypeScript not only enables developers to define a set of compatible values, but also to broaden this set or to narrow the set of compatible values as they go. The intersection type is a good example. In Listing 2, two types are defined: `Person` and `Studies`. The type `Student` is an intersection of both types. It includes all values that are compatible with `Person` as well as with `Studies`, which narrows the set of possible values.

This can be visualized easily through a traditional set diagram (Figure 1). A union type works in the opposite direction. Listing 3 defines groups of people that may be encountered at a university. This example does not claim to be complete.

The union set of `Student` and `Professor` is called `UniversityPeople`, and is a larger set than both subsets. There is an intersection, namely professors who study, but there are also compatible values that just fall into one of the two categories.

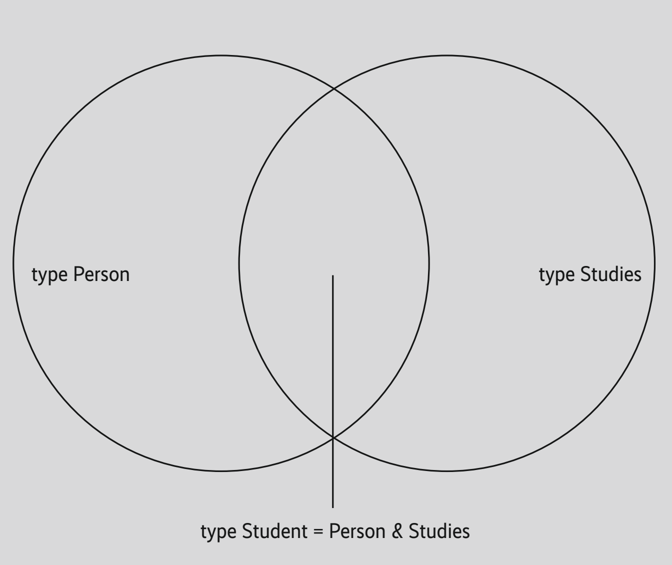
TypeScript's type system allows developers to define which values belong to which set. While union and intersection types are the most basic constructs it provides to

Listing 1: Structural Typing Looks at the Shape of Values, Rather than Their Name

```
type Person = {
   age: number,
   name: string
}

type Student = {
  age: number,
  name: string,
  semester: number
}

const student: Student = {
  age: 24,
  semester: 7,
  name: "Jane Doe"
}

const person: Person = student // this works!
```

Listing 2: Definition of an Intersection Type

```
type Studies = {
  semester: number
}

type Person = {
  name: string,
  age: number
}

type Student = Person & Student
```



>> The intersection type Student defines the intersection of Person and Studies (Figure 1).

define sets, they are just the beginning. TypeScript has much, much more up its sleeve.

Discriminated Union Types

The following example in Listing 4 shows further capabilities of TypeScript. In an exemplary toy shop inventory software, a data model is defined by using types. `ToyBase` describes all properties that are required by each toy. Afterwards, intersection types are used to define different kinds of toys in detail.

The use of intersection types is similar to the operation of extending interfaces. Both techniques narrow down the set of possible values by adding and requiring additional properties.

It is worth taking a closer look at the property `kind`. It is not a string as may be expected, but a literal string value. In TypeScript, each literal value can be a type. For example, the set of compatible values to the literal type `puzzle` is only `puzzle`, and nothing more. This might seem confusing at first, but has a huge effect on how TypeScript views a union type:

```
type Toy = BoardGame | Puzzle | Doll;
```

Listing 3: Defining a Union Type

```
type Professor = Person & {
    institute: string,
    employeeId: string
}

type UniversityPeople = Student | Professor
```

`Toy` describes all possible toys in the software. Using the `kind` property and setting it to literal values prohibits an intersection between the subsets. It is impossible that `kind` is both `doll` and `puzzle` at the same time. Therefore, values can be exactly one subtype. However, developers are able to discriminate between each type. This is how these types have received their name: discriminated union types. Discriminated union types can be used to write functions that provide exhaustiveness checks. This means they make sure that all possible variants of a union type have been exhausted, as demonstrated in the `printToys` function in Listing 5.

The switch statement enables iterating over each possible subtype of `toy`. In each case, TypeScript knows exactly which subtype to handle because of the uniqueness of the `kind` property. It gets even better: In the default case, TypeScript expects all possibilities to have been exhausted, which means the type of `toy` would be `never`, a type that contains no values. It is reserved for cases that should never occur—just like the default case in this example. If the default case does appear, that poses a serious problem as the data does not comply with the previously defined types. It is OK to throw an error here.

Additional safeties may be implemented, so as not to overlook new cases. For example, the union type `Toy` may be extended to include video games, as shown in Listing 6.

TypeScript will throw an error at the exact position where one might have forgotten to check for all cases (Listing 7). That is handy!

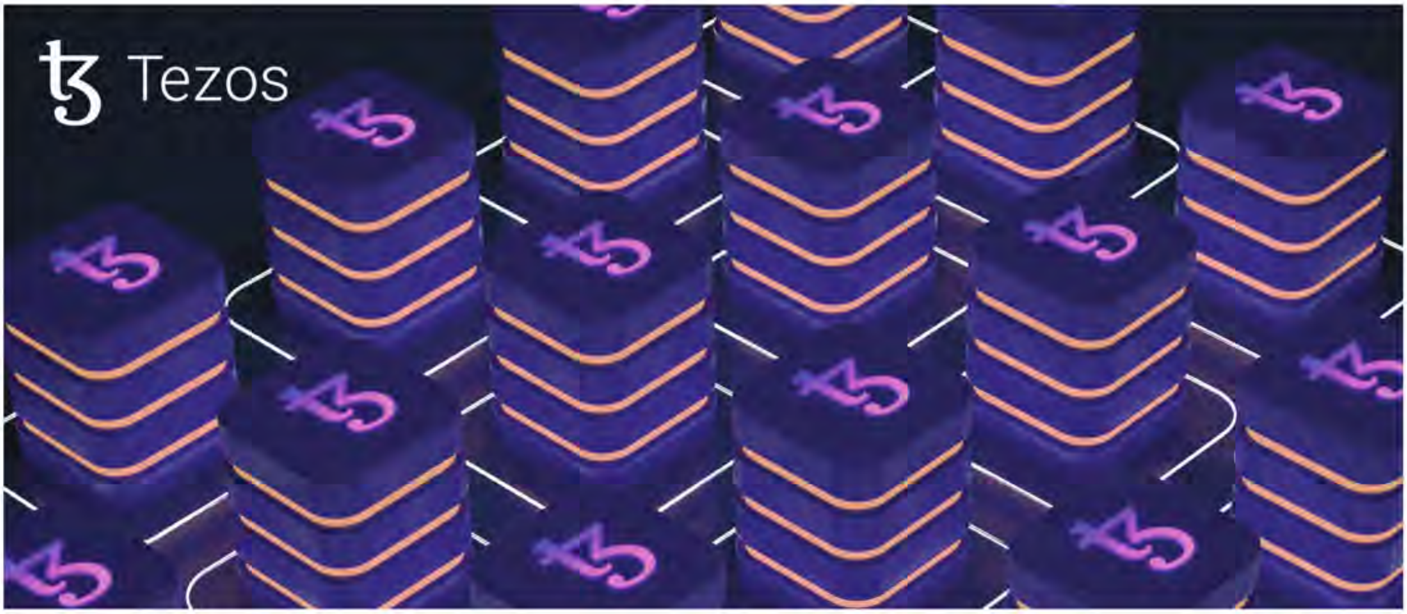
Mapped Types

A different part of the software needs a grouping of all toys. The properties of this type are equivalent to the values of the `kind` property in each toy.

```
type GroupedToys = {
  boardgame: Toy[];
  puzzle: Toy[];
  doll: Toy[];
};
```

`GroupedToys` may seem like an easy type but contains various possible pitfalls. For one thing, there are typos to consider. Nothing keeps developers from accidentally writing "boredGame" instead of "boardgame," which has the potential to break the code at some point. And with JavaScript, typos can happen anywhere!

Additionally, TypeScript has no clue that there is a relationship between the properties of `GroupedToys` and the `kind` literal type. This may lead to situations that require circumventing TypeScript's type safety by using explicit type asser-



Blockchain is the Foundation for Web3

This year, the most prominent developers from around the world join together once again at the We Are Developers World Congress in Berlin. The stage is set for lively debate and discussion about our technological future – new ideas about how we communicate, interact, and exchange value. But there is arguably no debate more consequential than the future of the Internet – and the blockchain technology that will be its foundation.

A wave of interest in the concept of a new Internet is building, as more and more of the building blocks have come into clearer focus. Web3 is about a lot more than just decentralizing Big Tech. Because blockchains don't exist in a single place, aren't controlled by a corporation or a state, and the data stored on them can't be changed after the fact, they have the potential to solve a lot of the problems created by traditional, centralized systems.

By design, Web2 is heavily centralized. Centralization has some benefits – organization and efficiency – that were critical in the growth stages of the web. However, such centralization deviates from the original vision of the Internet and has led to the key infrastructure of the web being dependent on powerful organizations and governments. Centralized infrastructure not only makes the networks less secure and easier to attack, but also sacrifices the censorship resistance that would be possible with a decentralized network. Users are subjected to the whims and wishes of governments and organizations that have full control over what information we have access to and what we're allowed to do or say – i creasing the power of authoritarian entities and limiting individual expression.

The hacker community that built the internet is being reborn in Web3 community across the globe, slowly realizing the true, original vision of the internet one line of code at a time – marrying efficiency and decentralization for a free and open web.

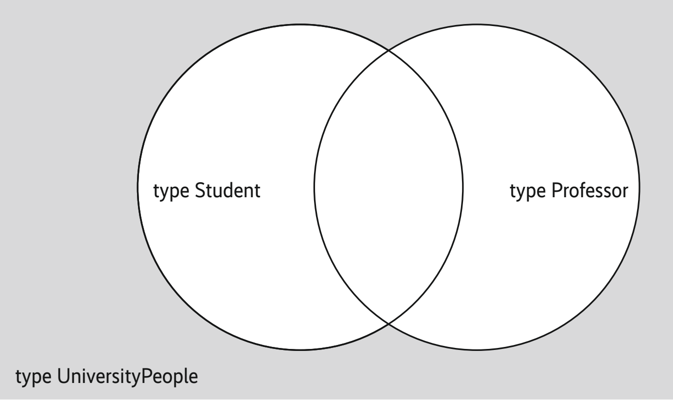
Web3 does not exist today in a complete sense. It is a fragmented dream of this new hacker ethos. A complete vision of Web3 will only be made a reality if this creative, liberated community continues to explore and build towards that enlightened vision – a fairer, more robust, and more private Internet, which will allow for the creation of a true 'metaverse' away from centralized control.

While lots of the noise around blockchain and cryptocurrency is focused on hype, in order to really understand the value of Web3 we need to look at value and practicality. Blockchains allow for the exchange of value between peers without the need to trust any centralized authority – and they can do so in a highly efficient and practical manner, being able to send transactions across the world in seconds, rather than days.

However, not all blockchains are created equal – and for Web3 to work, it requires a truly decentralized blockchain, controlled by a diverse community of users instead of centralized labs or VCs. Web3 requires a blockchain that moves at the speed of ideas, that is truly efficient, fast, and scalable in a sustainable way.

Web3 needs a blockchain that is designed to evolve and upgrade seamlessly, being built for impact and long-term utility, not short-term gains and hype That's why developers all over the world are choosing to build with Tezos, a pioneering Proof-of-Stake blockchain with a self-evolving system that prioritizes decentralization and community governance while enabling practical, scalable design.

From the very beginning, Tezos was built to avoid the problems of legacy blockchains, as a platform to build Web3 that's developer-led, by design. Create the building blocks of the internet of tomorrow on Tezos, the blockchain designed to evolve.



>> Union type, the union of Student and Professor (Figure 2)

tions and overrides. In many cases, developers end up maintaining the code. The moment `VideoGame` is added, it becomes necessary to update `GroupedToys`—another potential source of errors.

Defining relationships between types avoids this issue. To do this, it must be ensured that all toy kinds are covered. For this, a union of string literal types can be created directly from Toy by using an index access type.

```
// equivalent to "boardgame" | "videogame" | "doll"
| "puzzle"
type ToyKind = Toy["kind"]
```

Since the `kind` property has a literal type, `ToyKind` results in a union type with exactly four possible values. These values serve as keys for other values. Properties in JavaScript objects can be strings, numbers, or symbols. String keys can be anything in JavaScript, whereas TypeScript requires formalizing a subset. With ToyKind, such a subset has been established. Using a mapped type, this set can act as a basis from which to create a new object type.

```
type GroupedToys = {
  [Kind in ToyKind]: Toy[]
}
```

This type is equivalent to the original, hand-written `GroupedToys` type, but with an important difference: It has a relationship to `Toy`. Once `Toy` receives an update, `ToyKind` is updated as well, and so is `GroupedToys`. Therefore, maintenance only needs to happen in one place: the model.

Conditional Types

TypeScript allows developers to go even further with this. When generating new properties based on the union type `ToyKind`, the types associated with each property are a `Toy` array. Shouldn't more information be available, though? In fact, `GroupedToys` only makes sense if it can be ensured that each toy collected in `boardgame` is of the type `BoardGame`.

Listing 4: Using Literal Types

```
type ToyBase = {
  name: string;
  price: number;
  quantity: number;
  minimumAge: number;
};

type BoardGame = ToyBase & {
  kind: "boardgame";
  players: number;
}

type Puzzle = ToyBase & {
  kind: "puzzle";
  pieces: number;
}

type Doll = ToyBase & {
  kind: "doll";
  material: "plastic" | "plush";
}
```

Listing 5: Exhaustiveness Checks

```
function printToys(toy: Toy) {
  switch(toy.kind) {
    case "boardgame":
      // toy is BoardGame
      break;
    case "puzzle":
      // toy is Puzzle
      break;
    case "doll":
      // toy is Doll
      break;
    default:
      assertNever(toy)
  }
}

function assertNever(obj: never) {
  throw Error("I can't work with this type")
}
```

An exact mapping to valid types is possible. So far, the type alias `Kind` in the `GroupedToys` type has not been used to its full potential.

```
type GroupedToys = {
  [Kind in ToyKind]: Toy[]
}
```

Currently, it outputs `boardgame`, `videogame`, `doll`, and `puzzle` as new keys, but the same type can be used in different places. As a parameter for a custom generic helper type `GetKind`, for example. Quickly added to the code, `GetKind` is responsible for fetching exactly one subset of a discriminated union type based on the `kind` property.

```
type GroupedToys = {
  [Kind in ToyKind]: GetKind<Toy, Kind>[]
}
```

Shaping the Future with Ease

InterSystems IRIS for AI- and ML-Applications

Versatile application development
Embedded BI, AI and ML tools such as Adaptive Analytics and IntegratedML enable the immediate utilisation of data. Additional tools can be seamlessly integrated via APIs and PMML.

+

Excellent performance
Hybrid transactional/analytical processing offers more speed for multi-workloads in real time. The solution works faster than in-memory databases and guarantees efficiency.

+

Comprehensive data management
All relevant data is high quality and available at all times for AI and ML applications – regardless of source and file format.

The InterSystems IRIS data platform provides the perfect foundation to develop AI- and ML-based applications.

With InterSystems IRIS, companies easily develop innovative applications and services. Embedded Python enables AI and ML algorithms to run directly at the core of the data platform. At the same time, teams have direct access to numerous freely available Python libraries.

InterSystems®
Creative data technology

Interested to see for yourself? Test InterSystems IRIS now, easy and free of charge – scan the QR code and get started!

Listing 6: Adding a New Variant to the Toy Type

```
type VideoGame = ToyBase & {
  kind: "videogame";
  system: "NES" | "SNES" | "Mega Drive" | "There are no more consoles";
};

type Toy = BoardGame | Puzzle | Doll | VideoGame
```

Listing 7: Exhaustiveness Checks React Directly When the Original Union Type Has Changed

```
function printToys(toy: Toy) {
  switch(toy.kind) {
    case "boardgame":
      // toy is BoardGame
      break;
    case "puzzle":
      // toy is Puzzle
      break;
    case "doll":
      // toy is Doll
      break;
    default:
      // Wait! toy is of type VideoGame, not never.
     // Calling assertNever is not valid
      assertNever(toy)
  }
}
```

Listing 8: How Typescript Would Expand “A Conditional of Unions”

```
type Dolls = BoardGame extends { kind: "doll"} ? BoardGame : never |
     Doll extends { kind: "doll"} ? Doll : never |
     Puzzle extends { kind: "doll"} ? Puzzle : never |
     VideoGame extends { kind: "doll"} ? VideoGame : never
```

Listing 9: What GroupedToys Expands To

```
type GroupedToys = {
   boardgame: BoardGame[];
   puzzle: Puzzle[];
   doll: Doll[];
   videogame: VideoGame[];
}
```

Listing 10: A Generic Group Type for All Possible Models

```
type Group<Group extends { kind: string }> = {
   [Kind in Group["kind"]]: GetKind<Group, Kind>
}

type GroupedToys = Group<Toy>
```

It is possible to implement `GetKind` based on conditional types. They allow checking whether a certain type T is a subtype of another type U. This means that each value that is compatible with T is compatible with U as well. If that is the case, a conditional type returns the type in the true branch. Otherwise, it returns the type in the false branch.

`Extract` is a helper type based on conditional types. It does a subtype check as shown below: If type `T` is a subtype of `U`, it returns `T`. Otherwise, it returns `never`.

```
type Extract<T, U> = T extends U ? T : never
```

For example, `Extract` can be used to extract `Doll` from `Toy`. This requires a generic type `GetKind` that takes two generic type parameters. One is the group to extract from, the other one is the kind to extract.

```
type GetKind<Group, Kind> = Extract<Group, { kind:
Kind }>
```

```
type Dolls = GetKind<Toy, "doll">
```

`Group`, in this case `Toy`, is a union type. To TypeScript, a conditional type of a union is a union of conditional types. This means that a call like

```
type Dolls = GetKind<Toy, "doll">
```

which is replaced with

```
type Dolls = Extract<Toy, { kind: "doll" }>
```

expands to a union type like in Listing 8. Now every part of the `Toy` union will be checked against a type `{ kind: "doll"}`. Only one of the four types is an actual subtype of `{ kind: "doll" }`, namely `Doll`. Therefore, every value that is compatible with `Doll` is also compatible with `{ kind: "doll" }`. This condition returns the type in the true branch. In all other cases it returns `never`.

```
type Dolls = never | Doll | never | never
```

`never` is special in union types because it disappears. Since it is an empty set, this makes sense: If a developer creates a union of set `A` with the empty set, the result is set `A`. There are no new values added to `A` since, well, there are none to be added. Hence, in this example, only `Doll` remains.

```
type Dolls = Doll
```

Using `GetKind` in `GroupedToys` looks like this:

```
type GroupedToys = {
  [Kind in ToyKind]: GetKind<Toy, Kind>[]
}
```

`GroupedToys` has a dependency on `ToyKind`, which has a dependency on `Toy`. If `Toy` changes, so do `ToyKind` and `GroupedToys` (Listing 9). The result of these artfully crafted types couldn’t be more beautiful.

Using discriminated union types with a `kind` property is a common pattern in TypeScript. Therefore, a generic helper type to group something, no matter what kind of group, only makes sense. This is demonstrated in Listing 10. Thus, `GroupedToys` is zero maintenance: no typos, no branches to overlook. The model receives an update once, and the application knows what is happening.

Low Maintenance, High Productivity

TypeScript's type system is immensely powerful. Union, discriminated union, and intersection types allow for detailed modeling of data. Thanks to index access types, mapped types, conditional types, and generics, developers can create relationships across types. This brings TypeScript one step closer to describing any possible scenario of a JavaScript program with just a few lines of code. The type system does not get in the way, does not interrupt developers during their coding process, and only offers support by showing red squiggly lines where something might break.

If you want to know more about TypeScript and its type system, you can check out the author's blog (all sources for this article can be found here: ix.de/zs8y) and his book "TypeScript in 50 Lessons." Both deal with the long lasting aspects of TypeScript and explore the depths of the type system in detail. The book's website includes a sample chapter about union and intersection types. The TypeScript playground is a great place to start developing types in isolation. There you can also find all of the examples mentioned in this article. (mai)

Sources

All sources for this article: ix.de/zs8y

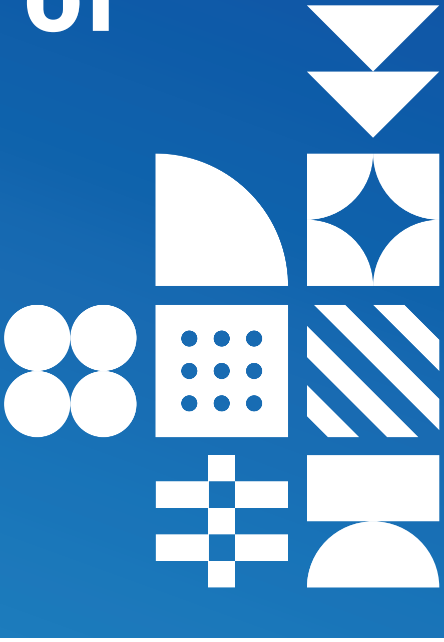
Stefan Baumgartner

works at Dynatrace. He writes for Manning and A List Apart, and made the Kessel Run in less than 12 parsecs. Recently, he has published his second book "TypeScript in 50 Lessons" with Smashing Magazine. In his spare time, he co-organizes ScriptConf, DevOne, and the local Rust meetup in Linz, besides co-hosting the German language Working Draft podcast. Stefan enjoys Italian food, Belgian beer, and British vinyl records.

> New Generation of Web Frameworks

Miško Hevery

Today's web frameworks are high in complexity and can lead to frustratingly slow websites. New requirements call for new web frameworks that leverage multiple CPU cores.



Developers use the web as a way to deliver content to their users. Most of that content comes in the form of web applications, and most of these web applications are built with one of many popular web frameworks. While these web frameworks are diverse in their APIs and philosophies, they are very similar in how they work and therefore alike in their limitations. Nowadays, these limitations are starting to show, and thus a new generation of frameworks is required to take applications to the next level. There are two basic things in building web applications that need to change. First of all, applications should take advantage of multiple CPU cores, and secondly, they should be able to stream content rather than simply load it.

The Problem at the Core of Web Development

The main problem of today's applications is that they have become very complex and contain large amounts of code that needs to be delivered and executed on the client. This causes long startup times and leads to frustration for users. In a way, web development tools are enabling developers to build more and more complex applications—but there comes a point when this is more than the browser can handle. Processors are hitting speed limits, and these days, more speed is achieved by adding more cores to the CPU, rather than by increasing the speed of single cores. Yet, today's web applications are built around a single-threaded model, which prevents them from taking advantage of additional CPU cores.

A typical site consists of two different parts—a first-party application as well as of code from third parties:

- First-party: This is a developer's own awesome application which they just built using the current-generation framework.
- Third-party: This is all of the code that is often needed to gain better insight into the application's users, e.g. Google Analytics, Google Ads, tracking pixels, HubSpot, Amplitude, A/B testing code, video players and customer support applications.

In a Nutshell

> Applications built with current-generation web frameworks often have long startup times due to their complexity and large amounts of (third-party) code.
> This calls for a new kind of web frameworks in order to increase page loading speed for users.
> There are some frameworks that follow a multithreaded approach and leverage web workers.

The car as a target for cybercrime?

Today's vehicles have a wide range of sensors, from ultrasound to radar, video and sometimes laser sensors, which act in conjunction with software and numerous driver assistance systems. All connected and online, designed to increase both safety and comfort while driving.

Dr. Martin Schmiedecker, Security engineer

Quelle: Bosch

When you buy a modern vehicle today, chances are high that it comes with multiple SIM cards included, and software updates are downloaded regularly. However, this makes cars potential targets for cybercriminals. To handle these threats, Bosch has multiple dedicated security engineering departments that deal with the cyber security of upcoming products. One of our security engineers at Bosch is Dr. Martin Schmiedecker. His job is to protect the systems, components and overall cars from malicious actors.

Secure from the start

"In the past, you had to break the window or pry open the door to get into a vehicle. Digitization and interconnection make it possible to gain control of a externally-owned vehicle or, in the worst case, even entire vehicle fleets, from a distance and even while driving," explains Dr. Schmiedecker. "The consequences can be fatal - from killing the engine to maneuvering the vehicle into a tree remotely." To ensure that such dramatic scenes are not realized and that drivers can trust the technologies, innovative companies such as Bosch place the highest value on the security of products and solutions: "Right at the beginning of the product development phase, I look at the security aspects, analyze possible threat scenarios and make recommendations on protective measures. Colleagues from different development teams then incorporate these into the software," explains the Bosch security engineer. These are then fed into build pipelines, tests are conducted automatically, and the entire software is run on real hardware.

Networked mobility of the future

Bosch is considered one of the innovation leaders in connected mobility. Driverless parking - an important milestone on the road to autonomous driving - is already a reality, thanks to Bosch and Daimler. One path of the future has already been mapped out: Cars will increasingly drive autonomously, the software of selected components is already updated over-the-air, and the navigation system searches for the destination address via voice input online. Experts refer to the car of the future as a "rolling data center". As a result, the complexity of the software and electronics in the corresponding control units and vehicle computers is also increasing dramatically. And in parallel, cyber security is becoming increasingly important. "For me, Bosch is at the forefront when it comes to implementing security measures in hardware and software," emphasizes Dr. Schmiedecker, who studied computer science at the Vienna University of Technology.

Staying on the ball with security

As a security engineer, you work on things that won't be driving on the road for another three to five years. The prerequisite for mastering this job is a technical degree from a technical college, university, or university of applied sciences, as well as enthusiasm for the subject.

> "It is immensely important to stay on the ball,
> to be always aware of the latest developments, attacks,
> techniques and countermeasures."

Interest in new developments is also indispensable. "It is immensely important to stay on the ball, to be always aware of the latest developments, attacks, techniques and countermeasures," the Bosch expert reinforces. "Only in this way can you weigh up the possible effects on future projects and take appropriate countermeasures."

Smartphone as a car key

One of Dr. Martin Schmiedecker's projects that will be available for many vehicle classes in the coming years is the smartphone as a car key. This is a vehicle-access system in which a separate car key is no longer necessary; instead, the key is stored securely in the smartphone. The car is opened and started with the help of the smartphone. Fixed sensors in the car recognize the owner's cell phone using modern cryptography, making it more secure than a fingerprint.

During the development of this project, international cooperation was at the highest level: Dr. Schmiedecker assisted the US client and programmers from Melbourne as a cyber security expert. He analyzed customer requirements and specified security measures, weighing how comfort and security could interfere with each other.

Bosch employs around 3,000 associates in Austria. At the locations in Vienna, Linz, and Hallein, Bosch operates international development competence centers for mobility technology - including Internet of Things solutions in the field of connected mobility. Technical talents will find an inspiring working environment at Bosch in Austria to actively drive future topics.

>> More information at www.bosch.at

All of the above code wants to load early and eagerly and as part of the main thread. The result is a flurry of activity on the main thread as the application starts up. This is such a big problem that Google has developed a specific PageSpeed Insights rating system for perceived site loading speed. This can be accessed on the web (all sources for this article are available here: ix.de/z424) and can be used to score arbitrary websites. The score is designed to encourage the industry to do better. The PageSpeed scores show the vast majority of web applications to be in the red, meaning the amount of code the browser needs to execute on startup is way too much to make for a good user experience. Some larger companies may be able to afford to allocate resources to this problem, in order to reach a yellow score. Green scores are almost nonexistent. Naturally, there are a lot of demo sites that show off how fast they are. Real production sites, though, which handle serious traffic and require a lot of third-party code but still manage a green score, are far and few between.

What Needs to Be Done?

The solution to the above problem is obvious, and yet very hard to achieve. The first step is to move all third-party code onto separate CPU cores, so that they no longer compete with the main thread for resources. In other words: Web applications need to become multithreaded. The main thread must stay empty so that it can deal with user interactions. The second step is to change the way of building applications so that they no longer require large initial downloads with long startup times. Ideally, as an application is being used, the framework downloads only the necessary code. Another way of looking at this is that the code is being streamed to the client in a just-in-time fashion, rather than delivering all of the code upfront.

Becoming Multithreaded

The programming model of the web was built on the idea of cooperative multitasking, where multiple codebases share the same single main thread, take turns and politely (operative word) yield to the next codebase when done. Recently, the web platform has gained multithread (and multiple CPU core) support through web workers. But adoption of this capability has been almost non-existent outside of a few cutting-edge demos. The main problem is that existing codebases do not run with web workers because they do not have access to the DOM and other bedrock APIs on which the applications and third-party scripts depend. Without support for the DOM, web workers tend to be of little use in practice. For example, Builder.io created the library Partytown. It aims to make web workers useful by recreating all of the APIs which third-party scripts expect from the main thread, while running with web workers. This allows the third-party scripts to run using a web worker, which in turn leaves the main thread with less work to do. It can thus focus on application startup and dealing with user interactions.

> Multithreading is coming to browser applications.

Finally, multithreading and multi-CPU are coming to browser applications. There are important caveats, though. Most importantly, code from a web worker will run slower than on the main thread. This is not an issue, however, as it is mostly used for third-party scripts such as analytics. It is likely that, over time, more libraries to run code in a web worker will emerge. This will make the process easier. Going multithreaded is a natural next step towards making the web faster, after all.

Partytown is a library designed to run third-party software via web workers. Integrating it into one's site is relatively easy as moving the execution of third-party code from the main thread to a web worker does not require any changes to the code itself. Only small changes are needed to move third-party code from the main thread to web workers. Creating applications that progressively load parts of themselves as the user interacts with them is a much more difficult task. This is because the process requires a change in the way of thinking about how to build applications, and thus requires a whole new breed of web frameworks.

Current-Generation Frameworks

All popular current-generation frameworks, while differing in syntax, basically work the same way. The mental model for these frameworks is as follows: The application code is the source of truth, and the DOM is something that the framework updates to match the application state. These frameworks have in common that, if a component is in the current render tree of the application, all of the component's code must be downloaded and run. Even if the component is lazy-loadable, and this is important, its inclusion in the render tree causes it to be downloaded.

The second similarity between current frameworks is that loading a component also loads all of the components' event handlers. These often contribute the bulk of the components' weight because most frameworks are synchronous in nature and therefore don't provide a place where code could be lazy-loaded, as that approach is asynchronous in nature.

WeAreDevelopers
#WEAREDEVS

Therefore, when the application first starts up, all components in the DOM must be downloaded, including their event handlers—regardless of whether it has been pre-rendered on the server or not.

The amount of work required to bootstrap or rehydrate an application is proportional to the complexity of the application that is being loaded. As applications become more complex, the amount of code the browser must execute on application startup is too high, precluding a good PageSpeed score.

Server-Side Rendering/Generation

Server-side rendering (SSR) or server-side generation (SSG) is a way to instantaneously show the application UI to the user. Using this method, the user is not able to interact with the UI until the application is bootstrapped as described above. So, while SSR/SSG is an important trick to make sites appear to load faster, they don't actually become usable earlier. On the contrary, it might be argued that the site is now actually slower because the browser has to download the application twice—once as HTML and then again as JavaScript.

SSR/SSG only makes the site appear to load faster, and the hope is that by the time the user figures out how to interact with it, enough time has passed that the JavaScript has been downloaded and can now be executed. It is important to note that the SSR/SSG content is not actually used by the bootstrapped application. If possible, the re-hydration process tries to reuse these nodes, but if they are missing, new nodes must be created, meaning the SSR/SSG content is in no way load-bearing to the application's functionality.

> **> The next generation of frameworks will be replayable.**

Frameworks: The Next Generation

It is the author's belief that the limit of what developers can achieve under the current paradigm, in terms of application startup speed, has been reached. This calls for a new way of thinking with regards to how an application is loaded. A way that does not require all of this code to be downloaded and executed eagerly. A framework that was designed from the ground up with SSR/SSG and lazy loading in mind is needed. In order to simplify the discussion, two new terms are coined: "replayable" and "resumable":

- Current frameworks are replayable. This means that when the application starts up on the client, it replays all of the work that the server has just done. This includes rebuilding the state, rendering trees, components, etc. Tricks can be used to lessen the amount of work that needs to be done, such as pre-serializing the state and reusing the DOM, but for the most part, the application has to re-execute from bootstrap every time, which is what makes it replayable.
- The next generation of frameworks will be resumable. This means that the framework will know how to serialize the state of the application during SSR/SSG, and deliver the UI in form of HTML, but also know how to serialize the application state in such a way that the application can resume on the client without needing to rebuild the state, rendering trees, components, etc. The ability of the application to continue where the server has left off is what makes it resumable. Resumability will make application startup extremely lightweight as the browser will have very little, if any, code to download or execute to resume the application.

In order to make applications start up faster, the browser must execute less code. The goal should be to execute little to no code, no matter the complexity of the application—therefore, resumability is important. With replayable frameworks, the amount of code the browser has to execute is proportional to the complexity of the application.

Resumability is not enough. It is necessary to consider how to break up current monolithic codebases into many small chunks and lazy-load them in a progressive fashion. A good mental model is an application which bootstraps as part of SSR/SSG on the server and seamlessly transitions to executing on the client, without having to re-bootstrap on the client. Furthermore, the code needs to be lazy-loaded on an as-needed basis. If the code has been loaded, it should be executed, otherwise it was loaded too early. The latter is the case with all of the event listeners which are currently loaded eagerly by most applications, even though the user may never interact with them at all.

Breaking Up Monolithic Code

Currently, there are no good tools available for breaking up large codebases in the desired manner. The only tool developers may currently use is dynamic `import()`. Dynamic imports are difficult for several reasons. The main one being that frameworks in use today don't support them as first-class citizens. At the core of the problem is the fact that dynamic import produces an asynchronous boundary in the codebase. If framework APIs expect synchronous functions, it is difficult to use an asynchronous primitive in a synchronous environment. There are many workarounds for async code loading in existence, but none of them are ideal.

For example, an application may have a button that is rarely clicked and therefore its handler should become asynchronous. The event handler API expects the callback function to return synchronously. This expectation that the event handler is synchronous makes it difficult to place a code loading boundary which is asynchronous. In practice, most developers don't judge this method to be worth the trouble and therefore it is often overlooked.

Both build tooling and frameworks need better support for asynchronous primitives. These are a must for lazy-loaded code. A mental model, and a corresponding framework that makes all of this seamless, is needed. Developers should be able to primarily focus on application functionality. The framework should naturally guide the developer to produce code that is easily broken up into smaller chunks and delivered to the browser on an as-needed basis. The opposite is true today. Currently, to follow a framework’s best practices results in a single code chunk. To break the application up into smaller pieces, more work has then to be done.

HTML-Centric

If the goal is to build resumable applications, HTML must be viewed as more than the framework's output. Instead, HTML code needs to become the source of truth. A source that enables application deserialization and continuation of its execution. HTML code must contain all of the information needed for the execution of the application to be resumed. These are some examples of what HTML should be comprised of:

- Listeners need to be serialized into HTML such that the framework can deduce which browser events need to be handled. Additionally, the code should indicate where the handle for the response should be lazy-loaded from.
- The state of the application needs to be serialized in such a way as to make it resumable.
- Components need to be reactive. Reactivity means that they need to track which data affects component rendering. Data tracking is referred to as subscription. The open-source framework Qwik, for example, can build up the subscription tree on the server and then serialize it into HTML. This allows the browser to instantly react to data changes without having to download all components to rebuild the subscription tree. Otherwise, the framework will have to rehydrate all components and re-render all of them, just in case the component depends on the application state which may have just changed.

This obviously requires a change in the way of thinking about HTML and what kind of information is stored there.

Can Existing Frameworks Rise to the Challenge?

Existing frameworks may theoretically be able to integrate these features into their way of thinking, but it comes at a huge cost. Making the framework resumable implies such a major paradigm shift that a project would have to introduce huge breaking changes to support it in its codebase. Those breaking changes would render the existing ecosystem unusable, and therefore it stands to reason that such a shift cannot be achieved by improving existing frameworks. What is needed is a fresh start with a focus on resumable first.

There are frameworks that have been built from scratch with these issues in mind, such as Qwik, an open-source framework we are working on at Builder.io. At its core lies the explicit goal of supporting resumable applications with fine-grained lazy loading. With Qwik and Partytown, we have been able to consistently achieve maximum scores from PageSpeed. The results are obvious, because the main thread workload is greatly reduced by using these technologies, and the sites feel much snappier.

Qwik is still in its infancy, but I believe it will be the first of many frameworks that focus not just on developer experience, but also on application startup. It is the only way to continue making applications more complex, while at the same time decreasing their bootstrap requirements. (mai)

Sources

All sources for this article: ix.de/z424

Miško Hevery

is the CTO at Builder.io, where he is helping to empower anyone to create blazing fast sites. Previously he has worked at Google, where he created Angular, AngularJS and was co-creator of Karma. Before focusing on making the web better, he brought testing culture to Google and the world through his blog. Miško started his career by designing digital circuits and moved on to databases, full-stack development, and finally, front-end frameworks, giving him a unique perspective. He understands all of the layers from the web down to a transistor.

> Increasing Gender Diversity in DevOps

Stefania Chaplin

DevOps is critical to the success of modern business because it facilitates everything from software development to IT operations. It remains to be a field with exceptionally low levels of diversity, and therefore monocultures are common.

It is the author's belief that businesses which address the issue of diversity in their DevOps departments will become more effective, as they start to disrupt orthodox thinking and embrace alternative approaches to problem solving. This not only improves their inclusiveness, but also productivity.

Different people tend to have different perspectives. In a group of heterogeneous individuals, a challenge may be approached in different ways, which will lead to a number of different solutions. This drives creativity and innovation. A mix of people makes a diverse set of skills available, which improves productivity and leads to better decision making.

The 2015 McKinsey report "Diversity Matters" (all links for this article: ix.de/z172) examined proprietary data sets for 366 public companies across a range of industries in Canada, Latin America, the United Kingdom, and the United States. It looked at metrics such as financial results and at the composition of top management positions and boardrooms. The findings speak for themselves: The companies in the top quartile for ethnic and racial diversity were 35% more likely to have financial returns above their industry mean, and those in the top quartile for gender diversity were 15% more likely to have higher returns.

In a Nutshell

- > DevOps, one of the higher paid roles in tech, has been shown to have low levels of gender diversity.
- > Diverse teams can help increase overall productivity and bring new ideas and perspectives to the table.
- > Companies may take different measures to increase gender diversity by attracting diverse applicants and retaining employees.

When companies look at customers who consume their products, they realize their customers come in different genders, ethnicities, ages, and have different life experiences. It is important to think about who makes the products consumed by the customers. Looking at the field of technology, such as social media companies, most contributors (from developers to CEOs) belong to a small demographic, whereas the product is consumed globally.

In the 2020 Stack Overflow Survey, over 68.3% of respondents identified as white or of European descent and 91.5% identified as male. The largest gender gap existed in DevOps, for which only 2.58% of the respondents identified as women. Their percentage increased to 7.67% for data scientists. Harvard Business Review analyzed the Stack Overflow survey and ranked the different jobs by attractiveness: Front-end developer and data scientist ranked near the top, backend developer and DevOps jobs came in near the bottom of the list.

In order to improve diversity in DevOps, companies can take three key steps:

- **Attract** a diverse range of top talent
- **Retain** employees
- **Develop** and promote individuals

DevOps is a growth market: There are more jobs than candidates in the field and it is one of the higher paid roles in tech. Improving diversity will also help with addressing other societal issues, such as the gender pay gap. To succeed, an organization needs to secure the best talent. Every role unfilled puts extra strain on existing team members, which might lead to burnout. Teams which aren't performing well cannot innovate, scale or automate their workflows.

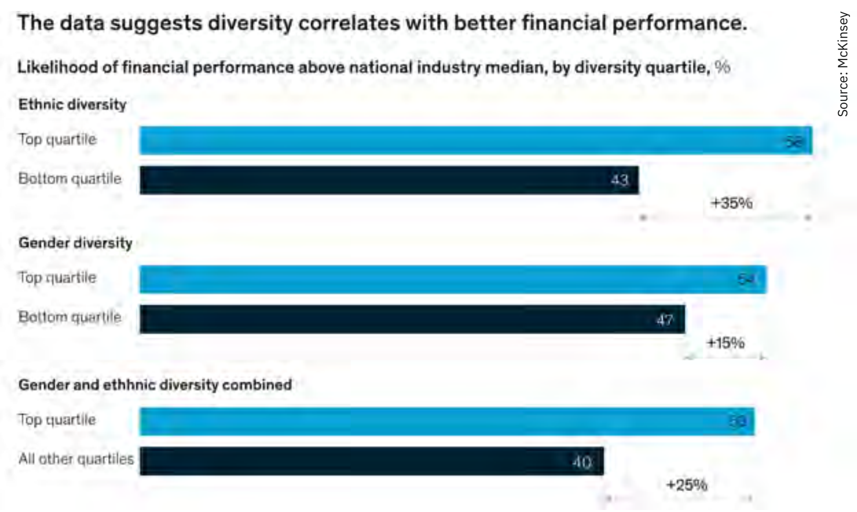
How to Attract Diverse Job Applicants

Job Descriptions: As a first measure, the hiring process deserves some attention. Hiring people works like a funnel. For each accepted offer, there may have been multiple prospective candidates, and each of those candidates may have had multiple interviews. Some candidates may not have been interviewed at all, and were rejected while screening applicants. Or they might have submitted an unsuccessful application. What about all the candidates who didn't apply because the job description was poorly worded? Companies may be missing out on a lot of fitting and diverse candidates due to the way they have phrased their job descriptions or where they have published their job adverts. Posting on the "Women who Code" job board, for example, may increase gender diversity.

When considering job descriptions, there are many different aspects to think about. Such as the wording that is used. Phrases like "coding jedi," "beer & pizza nights" and "hunting for exposed devices" can put (some) women off. It is essential to check the language of one's company for inclusiveness. Textio is a great place to start. This service analyzes documents and text snippets and makes suggestions. For example, "driven by" will become "inspired by." Phrases like "strong foundation" and "necessary change" are more likely to attract male applicants, while terms like "multi-faceted," "competitive position," and "informative" tend to attract women.

It is also worth considering who is writing the company's job descriptions. It is common practice to copy and paste these from a previous hiring process or from a different region and quickly publish them online. Studies have shown that job descriptions written by women tend to attract more women than those written by men. Language is important. This should not be taken as an invitation to increase the workload of the few women already working at a company. But, for companies struggling to attract female candidates, it may be worth giving these women a stake in the process.

Another question you should ask yourself is how the job description is structured. Let's take the example of a hire for a junior position. The job description may have ten bullet points. Typically, men who see themselves able to cover one or two of these bullet points would apply for the role, whereas most women who are able to cover eight or nine of them will say they are not confident in the last bullet point and thus hesitate to apply. This may be a sweeping generalization, but speaking from the experience of working in recruitment and



Source: McKinsey

>> McKinsey report "Diversity Matters," 2015: The graph shows financial performance, taking into account different types of diversity (Figure 1).

also coaching female friends from their first job through job moves and promotions, it hits surprisingly close to home. It may be worth thinking about whether experience with JIRA, REST APIs or Terraform is essential or only nice to have. Simply by breaking up the ten bullet points into five essential and five nice to have points, a rewritten job description can have a big impact on attracting female candidates.

The Interview Process: Ask yourself: Who is doing the interviews? Often, a combination of HR, a manager, and potentially someone more senior might be involved. There might be a peer review process and technical aptitude tests included. Depending on the role, the technical aptitude test might not be necessary. The applicants' technical ability might instead be ascertained by applicants sharing their portfolio from previous positions or by carefully interviewing them. Often, interview candidates are in full time employment at the time and have to juggle their personal life as well as their job and the interview process. Multiple interview stages and a laborious technical aptitude test can put off candidates.

> Job descriptions written by women tend to attract more women.

I previously worked in recruitment. During the hiring process for the job, I met three men, consecutively rising in seniority. At the end of the third interview with the Director he said, "I appreciate you've only spoken to men so far and we are very proud of our balanced team here, so I'm going to go upstairs and grab some of the women you will be working with." He didn't have to do that, but as a young female candidate it really made a difference for me! Even teams with little diversity may find other employees within the IT department who might be able to spare half an hour to meet a candidate. Most members of minorities will make the time to interview candidates in an effort to improve diversity.

Transparent Salaries: This is a tricky one. Transparency is key. If a candidate asks for a lower salary than was budgeted for the role, what does the company do? Depending on the socio-economic background of the candidate, their perception of a "high" salary may be very different from their peers'. In a viral tweet, a woman was thrilled about earning 100,000 USD at her new tech job in San Francisco, until she found out everyone else was making 120,000-130,000 USD. When hiring, it is important to note whether the company benchmarks salaries, both internally and externally. If the candidate is being offered equity, does the company offer an explanation on what this means? Not everyone will understand the nuances of options and RSUs. Providing a link to a third-party site explaining what these terms mean can help those who are financially illiterate or are too afraid to ask, because they are still in the negotiating phase.

Retaining Diverse Employees

Inclusive Onboarding: What does the onboarding process for the company look like? A company should make sure to build an inclusive onboarding experience. New hires should know that diversity matters to the organization and receive the relevant resources as well as enough space for settling in. Not everyone absorbs information equally, therefore companies should make sure the onboarding process caters for different learning styles. For a technical role, does each new starter have access to the correct tools, shared projects, and permissions? This is the first ideal from Gene Kim's "The Unicorn Project," entitled "Locality & Simplicity." It relates to the ability to get work done locally, e.g. effecting local code changes. If committing code requires approval from multiple departments, this process will create frustration and delays. Are processes in place to connect new hires with other DevOps colleagues to allow collaboration and inclusion right from the start?

Work-Life Balance: How is the employee's work-life balance defined within the organization? Diverse candidates may have different requirements in this respect, perhaps caring for younger or older generations within their family. It is one of the lessons from the pandemic that employees can be trusted to manage both their work and personal life while working from home. How do the DevOps teams in question handle remote work? Are they able to work asynchronously? If a very productive member of the team needs time for childcare one afternoon a week, can the company accommodate that? If that is not the case, why is that?

Assessing Job Roles: A good question to ask when assessing someone's position is whether the role is the right one for the person in question. It may be possible to create jobs that combine technical skills with skills from other business domains, if one takes the whole company into account. For example, when creating domain-specific solutions, using talent with a mix of skills can help with communication and collaboration between the two domains. Therefore, organizations should assess whether it is an option to create interdisciplinary roles or flexible paths to DevOps careers.

Is there an entry/associate level? There is no point in having an entry level role without the required support, as it would be doomed to fail. I joined the DevOps career as a Junior Developer. Then, at my next company, I was hired as Associate Solutions Architect. I was fortunate to have organizations invest in my development, which helped to accelerate my career to where I am now. Mentorship facilitates experiences such as these.

Supportive Mentorship: Mentorship can help to develop employees' skills, and expose and share knowledge within an organization. This accelerates the succession of leadership. For diverse candidates, mentorship is crucial. Mentorship has great benefits for both mentors and mentees. Both stand to gain from the shared knowledge. Within the context of DevOps, setting up pairs of employees with different technical skills can be invaluable.

I am fortunate to have had excellent mentors in my career who have provided a safe space for me to learn and ask questions. They allowed me to shadow projects, providing me with experiences I would not have had otherwise.

Utilizing Employee Groups: Women, LGBT, neurodiversity, Latinx and others—establishing employee groups creates a sense of belonging and comradeship. These groups can help hire diverse talent.

A personal highlight of my career was when, during the interview process, a female candidate asked her interviewer, an agile coach, what the women in tech stats of the company were and what the organization was doing about it. The agile coach explained that, although the stats weren't great, there was a group that met twice a month to discuss how to improve the stats and worked closely with HR to manage global recruitment strategy. The candidate said that this was the best answer she had ever heard and soon joined the company. Groups like these work best if they are established from the bottom up, so it is crucial for an organization to consult with its teams to discuss what they envision these groups to be like.

Giving Back to Communities: Is the company working with its local community? Meetups, universities or schools provide ample possibilities to do so. Working with the local community gives the organization external visibility and employees who contribute can use this as an opportunity to pay it forward. Meetup.com is one option to find local meetups. One of my favorites is the "Women in DevOps" meetup in London.

Career Development

Diverse Leadership: For external visibility, diverse representation in senior leadership is essential—be it as directors, C-level executives or on the board. Before joining a company, I will always look for evidence of women in senior leadership. If none is forthcoming, this is a red flag which I will bring up during the interview process. And I am not alone in this.

Of course, a business should avoid tokenism. No one wants someone appointed to a senior leadership position purely because of their gender or ethnicity. But there are steps that can be taken to compensate. For example, the business could have non-execs appointed to the board to ensure viewpoints from a wide spectrum are heard. In the same way, non-execs could be integrated into mentor programs to include viewpoints from across the business.

> Diversity leads to creativity and innovation.

Options for Promotions: As discussed, the gender divide leads to a difference in approach as to how people apply for a job. In a similar way, it can influence whether people feel confident about applying for a promotion. Therefore, line managers need to understand that a key part of their role is to identify people with the skills to become the next generation of leaders and to give them the confidence to advance if they want to.

It is important that managers speak to their team, as not all individuals will speak out if they want to be promoted. Some are happy where they are, some might need a nudge up the leadership ladder. Do employees in the organization have individual growth plans? Are they tracking their achievements? Encouraging this helps diverse candidates to rise through the ranks to senior leadership.

Developing Skills: Not everyone wants to be promoted. "Radical Candor" by Kim Scott establishes the model of superstars (want to rise to the top) and rock stars (stable as a rock). A good team needs a mix of both. What are the skills that individuals can work on? What is their goal/purpose in the organization? A goal like this can be outside of the day-to-day operation of the team. In "Radical Candor," a manager at Google uncovered that one of his employees actually aspired to be a Spirulina farmer. The employee learned that this meant working on their leadership skills, rather than their analytical skills that would have been required if their goal was to become a director at Google.

A good question to ask is: Does the company have a learning platform? Are employees encouraged to take the time to develop their skills? New technologies are developed frequently, which means it is important for the team to stay up to date.

Taking Action: Even if a business currently finds itself in a situation without a diverse workforce, there are immediate steps which can be implemented through an effective corporate social responsibility (CSR) program. The educational aspects of CSR can enlighten the workforce, exposing them to viewpoints and perspectives outside of their day-to-day. This investment into teams becomes a virtuous circle. Indeed, many of the leading HR consulting firms are advising that the next generation of star talent at a company are motivated by their bosses making sure they are working in a business that is seeking to be a positive actor on their environment. Therefore, if a business is struggling with its internal diversity, it can compensate through extensive CSR.

Diversity leads to creativity and innovation. It also improves inclusiveness and productivity. Every organization and each individual can help to attract, retain and promote a diverse range of talent by focusing on the three stages of "Attract, Retain, Develop." If we each improve each stage by 1%, through a compounding of the effects, this can help drive real change in an organization.

My Personal Experience

In my school year, there were approximately 100 girls. Humanities such as English, History and Philosophy were oversubscribed, while Biology and Chemistry had 30+ girls each, with some girls taking both subjects. In Further Maths, there were five of us (5%). When I went to university to study Computer Science, out of approximately 300 students around eight were female (2.7%). When I became a Python developer in a start-up, I was the only woman in the IT department. Then, when I moved into cybersecurity, I was the only woman in my global team of 15 employees. It was at that point, early in my career, disappointed by the lack of gender diversity and female colleagues, managers and mentors, that I decided, to inspire future generations, I needed to be the role model I had wanted to see myself.

I remember going to a company meetup and asking one of the product owners, "are there any women in your team?"

There were three others in all of Product & Engineering (4% of the team). During that company meetup, I founded the "Women in Tech" group for our company and spoke to one of the board members about how to improve diversity in our organization. My answer was: „visibility“—this is how we encourage future generations. During my tenure as President of the Women in Tech group, we increased women in technical roles from 4% to 11% in two years. Changes don't happen overnight. Making incremental and iterative changes, we were able to see actual results, including having at least one woman in each technical department. The final team to be joined by a female coworker was the Operations department. We celebrated with a party.

In summary, I do consider myself exceptionally fortunate. I mentioned my path through education, where my female peers at school mostly preferred humanities. What I didn't mention is the influence my family has had on me. My father, a qualified accountant and successful entrepreneur, was encouraging me with numerical-based games as soon as I could read, giving me confidence in my grasp of numbers. My mother, originally from South America, had built a career as a star broker in a major investment bank, making me believe I could succeed in business on my own terms as a woman. By the time I was 15 and thinking about A Levels, my older brother was already building his own technology start-up and encouraging me to pursue a STEM-based career. My older sister also organized a tech start-up in New York.

I believe that in most families, there is a gender bias even with the decision of what children should study. In truth, if we understand what we gravitate towards is what we enjoy, and what we enjoy is what we are good at, and that to become good at something requires an investment of time and energy, then, before we even know it, we are going to be successful.

My upbringing meant I felt supported in focusing on STEM-based subjects. I believe my childhood friends weren't encouraged in the same way, and therefore reverted to stereotype. If we really are to transform diversity in DevOps, we should re-think how we ask our children to engage with education. (mai)

Sources

All sources for this article can be found here: ix.de/z172

Stefania Chaplin's

experience as a Solutions Architect within cybersecurity, DevSecOps and OSS governance means she has helped countless organizations understand and implement security throughout their SDLC. A Python developer at heart, Stefania enjoys optimizing and improving operational efficiency by scripting and automating processes and creating integrations. She is an active member of OWASP DevSlop, hosting their technical shows. When not at a computer, Stefania enjoys surfing, yoga, and looking after her tropical plants.

> The State of Java

Maika Möbus

What is the current state of Java, which new features look exciting, and which are on developers' wish lists for future releases? Shortly before the release of Java 18, heise Developer interviewed nine experts about their usage of different Java versions, their experiences with upgrading, what new features they are most interested in, and what the future may hold.

heise Developer: The current LTS version, Java 17, arrived in September 2021, superseding Java 11 from 2018. Have you been using Java 17 in production, and if so, what was your main reason for upgrading?

Adam Bien: I immediately switched to Java 17 from Java 11. The main reason was new features like switch expressions and multiline strings. Java 17 is an LTS release—I would even update to Java 17 if the features were less useful.

Hendrik Ebbers: To be honest, I am using Java 8, 11 and 17 in production. Since I look after different customers and open source projects, there are different versions in use. For some of these projects, it currently makes no sense to upgrade from Java 8 to a newer version, because such a migration requires preparation and developer training. The good thing, though, is that all three versions are seen as LTS versions in OpenJDK and continue to receive security fixes. Thus, Eclipse Adoptium for Temurin will roll out Critical Patch Updates (CPUs) for all three versions within the next few years on a regular basis. However, it is important to make sure current updates of the LTS version in question are used in production. Of course, such projects must also plan a future migration and I recommend for every project to migrate to at least Java 11. But thanks to parallel LTS versions, this is possible in a planned and coordinated way. You should not be put off by the fact that Oracle JDK does not provide any (free) updates anymore for these versions. Oracle is definitely a special case in that they try to win commercial customers through security updates. Other vendors like Eclipse Adoptium are continuing to provide free updates.

Sandra Gerberding: Currently, we are still using Java 11, but will probably be switching to Java 17 soon. We always try to use the latest version.

Ivar Grimstad: I always upgrade to the latest version of Java as soon as it is released. Even non-LTS releases. I stay on the latest version of Java to get the latest language features, all the security updates and the performance improvements.

Tobias Modig: Unfortunately, we have not upgraded yet.

Sandra Parsick: So far, I have only upgraded a few projects to Java 17. Some of my projects are still running on Java

In a Nutshell

- > Java 17 is the current LTS version of the programming language. The latest non-LTS version, Java 18, was released in March 2022.
- > In this interview, nine Java experts weigh in on their experiences with Java upgrades, the shorter release cycle and new features.
- > Although not an LTS release, Java 18 brought some interesting features like the Simple Web Server and UTF-8 as the default charset.

8, and Java 17 is a good occasion to upgrade them—the main argument being that "it's about time." For projects running on Java 11, it is often hard to see a necessity to upgrade, although in those cases, it would be relatively painless. New projects often use Java 17 because it is the current LTS version.

Falk Sippach: Actually, I am not using any Java application in production, since I mainly work in an advisory function. With my clients and also with hobby projects, I have not seen any production application running on Java 17 so far.

Daniel Strmečki: Unfortunately, I haven't used Java 17 in production yet. I've only used it locally. Since my team is developing Java in an enterprise CMS framework, we need to use a Java version recommended by the framework vendor. Currently, this is still Java 11, but I would expect the vendor to upgrade to Java 17 soon.

Wolfgang Weigend: According to my ad hoc survey from the stage of the JavaLand conference 2022 during my JDK 17 LTS session to the audience of about 500 attendees in the room, around 50 percent are using the JDK 8 LTS in production for enterprise IT projects, followed by the JDK 11 LTS with an equivalent share of 50 percent in production. Only one raised his hand for JDK 17 LTS, and a few developers have already begun looking into JDK 18. Experiences regarding a stable Java basis platform with the necessary level of maturity allow the recommendation to use JDK 11 LTS in mission-critical projects in their production environments. In the meantime, I'm finding that companies are gathering their own experiences and tool migrations with JDK 17 LTS, which will lay the groundwork for a possible migration sooner than later.

Developer (architect), consultant, trainer (https://airhacks.io), podcaster (https://airhacks.fm) and Java enthusiast **Adam Bien** (https://adambien.blog) has been using Java since JDK 1.0 and JavaScript since LiveScript, and still enjoys writing code. He regularly organizes online live workshops (https://airhacks.live) and a monthly Q&A live streaming show (https://airhacks.tv).

Upgrading Experiences and Insights

heise Developer: Did the update process go smoothly or did you hit any bumps in the road? How were you able to overcome them?

Bien: There was no upgrade process. We switched the JDKs, set Maven to use Java 17, and then celebrated the success for a few seconds :-). We even "patched" the AWS Lambda environment to use Java 17. I expected some challenges here, but the upgrade went smoothly, even in this exotic case. All my projects are MicroProfile based, and all major runtimes, like Quarkus, Helidon, Payara, KumuluzEE, OpenLiberty, WildFly, and Piranha, have been supporting Java 17 for months.

Ebbers: I have actually made different experiences, depending on the client. Some of my clients were not able to upgrade at this time because some dependencies are not yet working with Java 17. Generally speaking, though, clients that had a clean upgrade to Java 11 have also had a smoother experience upgrading to Java 17.

Gerberding: Since we always try to use the latest version, we hardly run into any issues when upgrading. Larger version jumps can cause some problems though, especially when projects are still using old classes/methods that have already been deprecated, since newer Java versions have been cleaned up significantly.

Grimstad: My advice is to always stay on the latest version. If you do that, you will avoid the bumps in the road, and every upgrade will go smoothly. If you, on the other hand, only upgrade between LTS releases, you may have more challenges since more updates happen between them.

Parsick: For projects based on Java 11, the update has been going smoothly so far. Java 8 projects have run into some issues, but they were not specifically tied to Java 17.

Sippach: As I mentioned, I have personally not yet accompanied an upgrade to Java 17. Depending on size and initial condition (Java 8 or older/Java 11 or newer) of the project, the effort will be very different. From previous experiences, I do not expect there to be any great hurdles, though—especially when starting out with version 11.

Strmečki: Since Java is backwards compatible, the process of updating our code to the latest API is simple most of the time. It is worth checking removed and deprecated APIs in Java 12 to 17. Besides APIs, we might also run into build issues. Therefore, it is sometimes required to update and reconfigure build plugins.

Weigend: Due to its backwards compatible nature, Java allows for directly running older, not compiled programs on a current Java version. If new Java language features are in use, it is necessary to run the compiler and to modernize existing programs step by step.

heise Developer: If you or your company have decided to hold back on Java 17 and stick with Java 11—or perhaps even Java 8—for now, what are the reasons?

Bien: Sometimes developers are unaware of Java 17 being an LTS release, and they stick with Java 11 for reasons of inertia. After they've been told about the benefits of Java 17, most projects will be upgrading. These days, most workloads are containerized and therefore Java upgrades are less problematic than on bare-metal Java installations. With immutable infrastructure, you are packaging the JVM with the application. That makes it is easier to upgrade, and to roll back in case of any problems.

Ebbers: As mentioned before, I have different clients who are planning migrations in the future. Together with these clients, I have ensured that their Java runtime in production is pushed to the latest respective Java 8 update. Following this, we then plan the migration in a coordinated way. Often, I will go over the following steps together with a client: Initially, we check if the build process is set up in a clean way and would work with a newer Java version. It often happens that we restructure a Maven build a bit, for example, and introduce the Maven Wrapper. Afterwards, I work with the client to create a graph of their modules and dependencies. Now it is time to reduce the dependencies and update them to versions that are compatible with Java 11/17. And finally, the last step usually consists of implementing changes in the source code of the project. In all these steps, I try to involve the client as much as possible and to act as a kind of coach to make future migrations easier for them.

Hendrik Ebbers is a co-founder of Karakun AG. He founded the JUG Dortmund and gives talks and workshops on Java all over the world. His book "Mastering JavaFX 8 Controls" was published by Oracle Press in 2014. Hendrik is a JavaOne Rockstar, Java Champion and JCP Expert Group member. In 2019 he became a member of the AdoptOpenJDK TSC and the Eclipse Adoptium WG.

Sandra Gerberding has been a software developer since 2003 and has been focusing on continuous integration, test automation and deployment of Java web applications since 2009. In 2013, she started working as a DevOps engineer with a focus on CI/CD, Docker, Kubernetes, and cloud computing. Aside from this, she turns her attention towards life, the universe, and all the rest.

Ivar Grimstad is the Jakarta EE Developer Advocate at the Eclipse Foundation. He is a Java Champion and JUG leader based in Sweden. Besides advocating Jakarta EE technologies, Ivar is contributing to the Jakarta EE specification as well as being the PMC Lead for Eclipse Enterprise for Java (EE4J). He is also one of the specification leads for Jakarta MVC and represents the Eclipse Foundation in the JCP Executive Committee.

Gerberding: Within my organization, we always try to switch to the new version pretty quickly. But currently, it hasn't been going as fast for internal reasons.

Grimstad: If I was employed at a company that refused to upgrade, I would probably just look for another job.

Modig: Moving to Java 11 was pretty time-consuming for us. There is so much more to do than just changing the version in the pom-file. It affects everything, from upgrading dependencies, application servers and frameworks, to testing and documentation. It's the usual battle between doing technical improvements and building features for our customers. So far, building features has had the highest priority, but we will get there in the end.

Parsick: The main reason is always time. Usually, we roll out an update when external circumstances require it. Projects that are still using application servers are sometimes even forced to stick with Java 8 or 11 because the application servers don't officially support Java 11 or 17.

Sippach: The Java world seems rather conservative to me, and I believe that is why only a few companies will upgrade to the new version in a timely manner. There are numerous reasons for this, in my opinion. On the one hand, all dependencies (libraries, development tools, etc.) must support the latest version. On the other hand, older Java versions always receive support for a long time. Security vulnerabilities or serious bugs are therefore not to be expected, and these versions are usually especially stable and their feature set is well-rounded. In my experience, nobody is in a hurry to update existing software to the latest version. However, when developing new software, using the latest release is common practice. By the time a new project or product goes into production, there usually have been a few new releases anyway.

Strmečki: I would suggest not to update to a new LTS version immediately. Instead, it is better to wait for a few months until all plugins and IDEs have caught up with it. A few minor fixes for performance or security issues will probably also be out by then. Right now seems like a perfect time to update to Java 17, for example.

Weigend: From my personal project experience, it depends on the stage of application development and any dependencies the team may have with their toolchains. Currently I recommend first considering the JDK 11 LTS release for enterprise critical applications, but given the quick adoption of JDK 17 LTS, it is already proven and mature so it may be a good option—especially for organizations looking far out for support requirements.

Feature Highlights in Java 17

heise Developer: Java 17 added some new features with Java Enhancement Proposals (JEP) like "JEP 406: Pattern Matching for Switch (Preview)" and "JEP 409: Sealed Classes." What are your personal highlights?

Bien: "Pattern Matching for Switch" is a useful feature for processing data, for example JSON-P types. This feature is in preview, so we are not using it in production yet. We are not using inheritance in cloud-native/stateless/serverless microservice projects. Sealed Classes are less appealing in the cloud/stateless context, but they provide additional documentation at the interface level. The interface already lists all implementations that introduce a pragmatic opportunity for quick code navigation:

```
sealed interface Celestial
   permits Planet, Star, Comet { ... }
```

By switching to Java 17 directly from Java 11, "JEP 395: Records" (Java Records) became immediately available for production projects. Records are a "one-liner" immutable data container. I use Records everywhere: from JSON serialization over a concise method parameter to a classic Data Transfer Object replacement.

Ebbers: Instead of a new language feature or API, I would rather focus on aspects that have been introduced behind the scenes. Specifically, I would like to point out the move of the OpenJDK source code to Git and GitHub. Such changes are also reflected in OpenJDK as JEPs: With "JEP 357: Migrate from Mercurial to Git" and "JEP 369: Migrate to GitHub," the migration to GitHub was finalized with the release of Java 16. This move makes it much easier for the community to contribute to OpenJDK, since widely known and well documented tools and workflows have now become available. The migration has already resulted in some members of the German Java community submitting code contributions to Java.

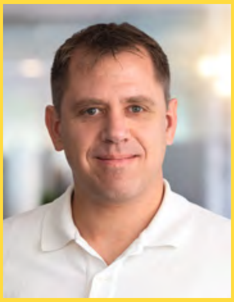
Gerberding: Pattern Matching for Switch is definitely a highlight for me, as well as Records that arrived with Java 16.

Grimstad: I don't really use switch much in my code, so Switch Expressions are a nice addition, but not that important to me. My personal favorite among the recent additions in Java is Records. They were introduced as previews in Java 14, but became fully integrated in Java 16.

Modig: Pattern Matching is definitely something that I look forward to. For me, Pattern Matching has been one of the biggest features of Scala and similar programming languages that I was jealous of, so it will be nice to finally have that in Java.

Parsick: My personal highlight is the introduction of Records (JEP 395). Strictly speaking, this is a feature of Java 16 (not taking into account preview releases). But I believe this feature will be used in more projects with Java 17. This will further reduce the arguments for using libraries like Lombok.

Sippach: I am dealing with the new features mainly in my talks and articles, but have also been using them in small demo projects. What currently excites me the most is Pattern Matching, although it is far from finished. For me, the features that have been gradually introduced since Java 12 are part of this, such as switch expressions, type patterns (Pattern Matching for `instanceof`), Records, Sealed



Tobias Modig is a developer, coach, speaker and teacher, who values clean code and good coding habits just as much as a positive work atmosphere. He has been working as a developer and software architect since the late 90s and is currently a software development consultant at Citerus in Stockholm.

Sandra Parsick is a Java Champion and works as a freelance software developer and consultant in the Java environment. She has been dealing with agile software development in various roles since 2008. Her primary areas of expertise are Java Enterprise, cloud, and the automation of software development processes. She enjoys writing articles and giving talks on these topics.

Classes, and especially Pattern Matching for Switch, which is currently still in preview. Most of these features may be used independently from Pattern Matching. In the years to come, I believe we will learn to specifically appreciate Records. They allow developers to create data classes in a light and compact way and the compiler will take care of creating the otherwise tiresome boilerplate code.

Strmečki: Pattern Matching has really improved the usability of the switch statement. Patterns in switch enable much finer case selection. However, it’s still just a preview feature, so probably most developers won’t use it yet. Personally, I don’t think I will be using Sealed Classes that much. I see this feature as more useful for developers that work on the development of frameworks and common libraries. My personal highlight are Text Blocks and Records. I think those two features will be used most by developers.

Weigend: In JDK 17, my personal highlight is "JEP 412: Foreign Function & Memory API (Incubator)." This API allows Java programs to interact with code and data outside of the Java runtime. Through efficient calling of foreign functions, with code outside of the JVM and through secure access to foreign memory that isn't managed by the JVM, the API allows Java programs to call native libraries and to process native data. This avoids the brittleness of Java Native Interface (JNI).

Falk Sippach works at embarc Software Consulting GmbH as a software architect, consultant and trainer. For over 15 years, he has been supporting mostly agile software development projects in the Java environment. As an active member of the Java community, he enjoys sharing his knowledge in articles and blog posts and as a speaker at conferences.

More Frequent LTS Releases

heise Developer: With the release of Java 17, Oracle announced that it intends to release long-term support Java versions every two years instead of every three. The next planned LTS version is Java 21, set to be released in September 2023. What is your take on the shorter release cycle and could this lead to companies skipping certain LTS releases?

Bien: I like the bi-yearly releases. Even a yearly release would work for most companies. Today, software is updated more frequently; a Java update can be incorporated into the regular release cycle.

Ebbers: In general, I think it's a good idea, and I believe that the companies working on the OpenJDK can do this work and provide security fixes for the more frequent LTS versions. However, LTS versions with a quicker release cycle must also be supported by the community with frameworks and libraries. I have already heard some critical voices of leaders of different projects, saying that this will further increase the workload within open source projects. Adding to this, Oracle has unfortunately done a catastrophic job in terms of organization and press when it comes to this new development. The information has not yet reached nearly as much of the community as it should have, and this will surely lead to numerous problems and mismatches. Many important players in the OpenJDK community only found out about this change in the release cycle through the announcement, and thus were not able to give their input beforehand. Obviously, Oracle as OpenJDK Lead must learn to communicate better and more openly with the community.

Gerberding: In my opinion, this is a good decision as it will make updating from previous versions easier. I hope that it will encourage large organizations to start upgrading to newer Java versions as well.

Grimstad: I don't really care about the LTS versions of Java. I think you should always stay on the latest version. But if the policy is to only upgrade between LTS versions, I think a shorter cycle is good. Skipping an LTS release would be a really bad decision, so I would probably recommend to anyone working for organizations doing that to follow my recommendation in question 3.

Modig: The shorter, the better. In my experience, many (especially bigger) companies only allow us to move between LTS versions, which means developers often need to wait for years to have that nice new feature. Every attempt to shorten that waiting period is more than welcome.

Daniel Strmečki works as Director Digital Platforms, Adobe at ecx.io, which is part of IBM iX, in Varaždin, Croatia. In early 2019, he earned a PhD in Information and Communication Sciences from the University of Zagreb. In his spare time, he writes articles on baeldung.com, one of the world's most popular Java blogs.

Parsick: I think this will have little impact on my projects. Perhaps, organizations will start to use several different Java versions in parallel because of this.

Sippach: The reasoning behind this step is rather unclear to me. It probably has to do with the fact that Oracle started charging a fee for its Oracle JDK in production a few years ago, which had been free until then. Because of this, it now has competition by some alternative distributions of the OpenJDK. Temurin from the Adoptium project (previously AdoptOpenJDK), for example, is supported by many vendors, receives support for a very long time and is available for free. Since version 17, the Oracle JDK may be used in production for free again, and perhaps the shorter release cycle is meant to draw attention to this. Creators of Java tools and libraries are moaning, though, since their release cadence is also being shortened to two years. On the other hand, two years mean less time in which things can happen that would justify upgrading to every new LTS version as it comes out.

Strmečki: Personally, I like the shorter release cycles. They provide us with more options and flexibility. Thus, we can choose when to update and to which version. It's always good to stay up to date, so I don't see a reason why we should skip specific LTS versions.

Weigend: The release of JDK 17 was a good occasion to contemplate the shorter release cycle that was introduced a few years ago and make possible corrections regarding the future course. Developers and organizations have had time to switch from the model of "massive changes every few years" to the better digestible version rhythm of six months. While several developer surveys show that between a quarter and half of the participants use the six-monthly non-LTS releases, only about half of them stated to be using them in production. Toolchains have adapted to the most common developer tools and frameworks and quickly added support for new six-monthly Java releases, sometimes even before their actual release date. Although the six-monthly usage is on the rise, the feedback is clear: Most organizations still prefer using LTS releases in production.

With this in mind, it seems to make sense to switch future LTS releases to a two-year rhythm. This will provide organizations and developers with more possibilities to have an application run on an LTS version, knowing that it will be supported long-term with the necessary stability, security, and performance updates. Using six-monthly releases is attractive because organizations are aware that the next available LTS will always be less than two years away. Every LTS version is planned to receive long-term support for a minimum time frame of eight years in order to allow for enough time and opportunity for organizations with a Java SE

Subscription to migrate between the Oracle JDK LTS versions as desired.

A Look at Java 18 and Beyond

***heise Developer:** Is there anything in the current non-LTS version Java 18 that has sparked your interest?*

Bien: I already used the built-in HttpServer for testing purposes or to serve static pages several years ago with Java 1.6 (all links for this interview are available here: ix.de/zqtt). The internal web server functionality is becoming available as a CLI (jwebserver) and an API now. The "JEP 408: Simple Web Server" could become a helpful tool for tests and development. Java 18 will also set the internal encoding to UTF-8 ("JEP 400: UTF-8 by Default"), which could save some debugging time. I'm especially looking forward to "JEP 413: Code Snippets in Java API Documentation." With Java 18, Javadoc gets new capabilities for displaying code snippets. Example code always was a part of good Javadoc comments since JDK 1.0, so the Code Snippets will improve the overall quality in Java projects.

Ebbers: A list of all JEPs in Java 18 is available on the website of the OpenJDK project. For me, a small highlight is JEP 408, which introduces the Simple Web Server. By using the new jwebserver command line tools or the new Java API, I can very easily start up a web server.

Gerberding: I find JEP 408: Simple Web Server quite interesting. I like to have small tools available for quickly testing things outside of the project context.

Grimstad: Java 18 will specify UTF-8 as the default charset (JEP 400). This is a good thing. Other than that, I find the possibility to include code snippets in JavaDoc (JEP 413) the most useful feature of Java 18.

Modig: "Sparked my interest" might be too strong—but I believe that "UTF-8 by default" will save a lot of time. People still run into encoding issues on a day-to-day basis and if we could reduce those issues by, say, 25 percent, it would be a huge amount of time saved, worldwide.

Parsick: The second preview of Pattern Matching looks interesting. It is nice to see UTF-8 being defined as default charset.

Sippach: Yes, definitely. For example, the further progress of Pattern Matching for Switch. In version 18, I would have wished for Deconstruct (Record) Patterns to be introduced, but unfortunately, they have not made it into the release. I am now hoping to see them in Java 19. Furthermore, I like that Java 18 has introduced a unified default charset across all operating systems—always UTF-8. The Simple Web Server is a nice idea as well. It is also exciting that Java is supposed to continue to become less cluttered, and in line with this, the finalization process has been deprecated and will be removed in the future.

Strmečki: Not really, to be honest. From what I see, there won't be any major changes in the API in Java 18. Patterns in Switch that will go into a second preview, while UTF-8 will become the new default charset.

Weigend: Yes, the Simple Web Server (JEP 408) integrated in JDK 18, a minimal web server for the command-line and other areas of usage like education, looks interesting. It is an out-of-the-box static HTTP server with a dedicated command-line tool and a new API that was developed for prototyping, tests, and debugging. I find it noteworthy how the tool jwebserver.exe and its interfaces can help dealing with routine tasks and special cases.

heise Developer: Moving forward, what features would you like to see implemented in Java in the future?

Bien: I'm already looking forward to all the previews being available as GA features in upcoming Java releases. Synergies between existing, or slightly extended, JEPs will provide a significant benefit. Combining "JEP 406: Pattern Matching for Switch (Preview)" with "JEP 409: Sealed Classes," "JEP 394: Pattern Matching for instanceof," and "JEP 395: Records" might introduce an entirely new "object destructuring" capability—the ability of extracting only interesting information from an existing object instance. Object destructuring would further streamline many CRUD applications. Also, further iterations on the "JEP 378: Text Blocks" could introduce native String-templating capabilities to Java. The current multiline String is already heavily used in my projects. An extended version of Text Blocks would make complex templates easier to maintain. All the recent JEPs have laid the foundation for a bright Java future. I expect to

Wolfgang Weigend works as a Master Principal Solution Engineer at Oracle Global Services Germany GmbH in the worldwide Java team. He works with Java technology, GraalVM, and architecture for enterprise-wide application development.

get major features faster in upcoming Java releases by combining the already existing language features.

Ebbers: From a technical point of view, I am very pleased with Java. However, I think there is still a lot to do in the ecosystem. The migration to GitHub has simplified contributing to the OpenJDK, but aspects such as the OpenJDK bug tracker and the OpenJDK website remain incredibly outdated and are in urgent need of an overhaul. In my opinion, the bug tracker also requires new workflows in order to better include the community. I have recently published a long report on my experience with it (all links for this interview: ix.de/zqtt).

Gerberding: It would be great if Java increasingly offered the possibility for functional programming.

Grimstad: Nothing in particular at the moment.

Modig: Project Loom—for us mere mortals not living on the beautiful island of Crete, concurrency and threading are still cumbersome and heavy. If threading could be made more lightweight and easier to use, the Java community would gain a lot.

Parsick: Spontaneously, I can't think of anything.

Sippach: I myself have no ideas where Java should go from here. After having been around for almost 30 years, the language is more than mature now. I do like that it has been continuing to develop in small steps, as well as further following the functional style with pattern matching. Additionally, some interesting topics like Value Types and Fibers (lightweight Threads), that I am really looking forward to, are in development.

Strmečki: I think it would be cool to introduce a keyword for creating utility classes in Java. That way we could ensure that utility classes contain only what they are supposed to contain, namely static methods. Additionally, we could avoid those annoying warnings that force us to add a private constructor to utility classes.

Weigend: I would like to see the contents of Project Panama put into practice, for example to accomplish a move away from the Java Native Interface (JNI). It used to be the only option to access native libraries from Java code, but its susceptible and complex programming mode have made it hard to use in a larger scope. But that is currently changing. The APIs and tools of Project Panama are working together to create a secure, modern, and efficient alternative for accessing foreign memory and external native code from Java.

heise Developer: Thank you for the interview!

The interviews were conducted by Maika Möbus. (mai)

Sources

All links for the interview including the mentioned JEPs can be found here: ix.de/zqtt

Application Development in the Age of Digital Transformation

The transformation of application development and the associated creation and operation of cloud-native application development platforms confront companies with various challenges.

The implementation of new processes and technologies, especially in application development, is an essential component of the digital transformation journey in achieving a company's competitive edge in the marketplace. anynines offers companies tailored and automated cloud platform solutions. The goal is to support companies in the emerging challenges to make this journey as efficient as possible.

Flexibility through Modularization

With a9s Platform – a solution that simplifies the creation and operation of application development platforms – anynines addresses this challenge.

The modular and automated approach of a9s Platform enables companies to customize the selection of individual components that take care of both applications and data to their exact business needs. This saves time and resources which can consequently be invested in other areas.

Automation and Independence

The modules, all designed to run on Cloud Foundry or Kubernetes, are Application Runtime or Container Runtime, industry-proven Data Services, and Platform Operations if required. With the included automation, building application development platforms is simplified. Platform operators reduce the operation of one or more platforms to the bare essentials. The strict separation between Platform and underlying infrastructure that a9s Platform provides, allows application development platforms to be deployed on any major infrastructure.

This keeps it vendor agnostic. Platform Engineers and DevOps at anynines are constantly working on expanding the solution to deliver an even better user experience of the a9s Platform. Therefore, our growing teams are continually looking for reinforcement, especially with Kubernetes experts.

For more information about
the a9s Platform, the team
behind it, and open positions,
please visit our website:
www.anynines.com

> How Your Bundle Size Affects the Climate

Roy Derks

Have you ever thought about how climate change affects your work as a developer—and vice versa? As it turns out, there is a way to build more climate friendly websites.

Climate change is happening all around us and is getting more evident. To try and stop this process, various measures are being taken. In many countries, the transition from fossil fuel to green energy has begun, and electric vehicles are rapidly taking over the streets. As a developer, you can contribute to slowing down climate change, not only by using green energy or by driving electric. As people worldwide are using the internet daily, the websites and applications they visit and work with have become a large part of their energy usage. Here, you can make a difference with your code, too.

Global Internet Usage and Website Size Are Increasing

Global internet usage is increasing rapidly, and subsequently, the time people spend online is increasing as well. People use the internet for various reasons, such as work, social connections, and entertainment. Desktop computers are no longer the main device for browsing the internet, as mobile phones have become the most important screen for many users. The Digital 2022 report (all links for this article ix.de/zpuu) shows that 92% of almost five billion people using the internet globally do so on their mobile phones. With the internet in one's pocket, it has never been easier to access and visit websites and use applications. On average, people spend seven hours per day on the internet, a 4% increase from the year before.

There are multiple reasons for the increase in average daily online time within the last ten years, but some of them are not as obvious as one might think. For example, most people spend the majority of additional time on the internet doing work rather than in their free time. Also, new trends have led to increased usage next to more straightforward access to the internet. According to the Global Web Index, over three-quarters of people between the ages of 16 and 64 shop online every single month. This includes increased demand for online grocery shopping, which almost doubled in 2021.

This poses an issue because data needs to be downloaded from the internet onto the (mobile) device with every visit to a website, which requires bandwidth and, ultimately, a data center at the backend. According to the HTTP Archive's data,

In a Nutshell

- > Websites have grown larger, more complex, and more energy-consuming, thus affecting the climate.
- > Developers can take different measures to make their website more climate friendly.
- > Choosing energy efficient programming languages, reducing JavaScript bundle sizes, and switching to green data centers can make a large difference.

the size of the average website was four times larger in 2021 than in 2011. Data for websites can be a heavy burden for countries with data centers. In the Netherlands alone, data centers account for more than 4% of the total electricity consumption. Luckily, these data centers are increasingly using green energy and are collaborating with data centers in the rest of Europe to become climate neutral by 2030.

Websites Have Changed

Ten years ago, there were much fewer websites than there are today. The demand for websites has increased among all types of businesses, including people that wish to create a personal site. People who had a website ten years ago were likely to have a .com, .net or .org extension for their domain names. Today, there are hundreds of other domain extensions to choose from. Extensions like .io, .dev or .ai are popular among developers who are building a website for their business.

The most popular websites ten years ago included Facebook, Google, and YouTube. Although these still remain among the most popular sites, their content has drastically changed, along with their applications. Due to technological development, these websites include images and videos of higher resolution than before. Images and videos are also the main content of the most common applications in recent years.

The technologies used to create applications for the web have changed greatly. Current web applications are largely built with JavaScript. The Stack Overflow Developer Survey of 2021 shows that 65% of all professional developers use JavaScript. It can be used client-side and server-side. Also, it is much more common to find a popular JavaScript framework being used instead of vanilla JavaScript.

These frameworks help to extend HTML and CSS, allowing developers to easily create complex web applications. They produce cleaner and more concise code, as well as facilitate sharing components across different projects with greater ease. This process started with the JavaScript library jQuery, followed by the frameworks Angular and React. And these are just the most popular ones. There are hundreds of JavaScript frameworks for creating aesthetically pleasing and highly functional websites.

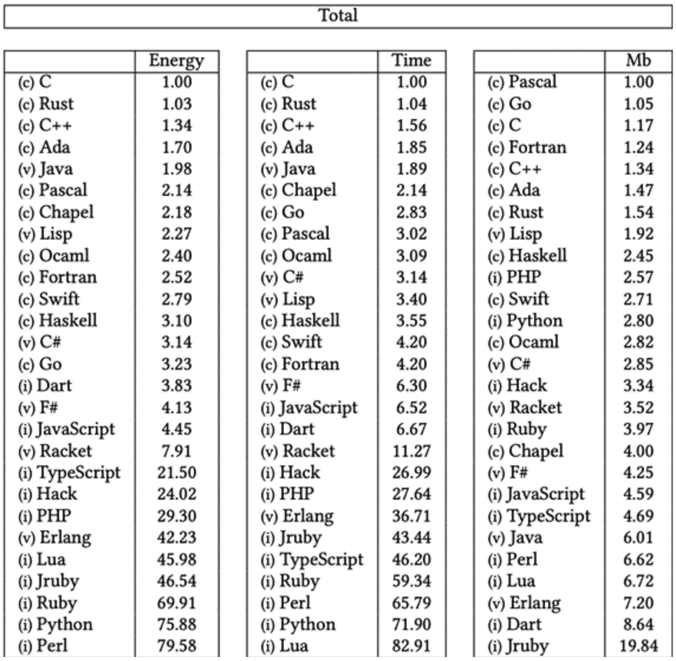
In the author's experience, the size of the JavaScript bundles that are generated by these frameworks is often around 2 MB, but can reach 4 MB or more. The bigger the bundle

for a website, the bigger its impact on the climate. With a bundle size of around 5 MB, the climate impact of a website such as Website Carbon with 10,000 monthly pageviews can be the same as driving 10,000 kilometers with an electric car. According to calculations by Website Carbon, each visit to such a website may produce around 5 grams of CO_2. This does not include the images and videos that are loaded separately from the website bundle. JavaScript is known to be a programming language that consumes a lot of energy, so not only the bundle size of a JavaScript application matters.

Energy Consumption of Programming Languages

Not only has the number of websites changed, but also the way they are built. In the most recent years, the yearly Stack Overflow Developer Survey has found JavaScript to be the most popular programming language amongst developers, which has led to increased bundle sizes. In comparison, ten years ago, SQL was the most used programming language. JavaScript is a scripting programming language, meaning it has other flaws as well. Scripting languages generally consume more energy than other types of programming languages, because they include a higher level of abstraction from the bare metal to make the life of the developer easier.

Going back to the Stack Overflow Developer Survey, aside from JavaScript, both TypeScript and Python have seen an increase in usage. Python has been gaining popularity in favor of C#, which is a low-level imperative programming language. The energy consumption of Python compared to C# is around 25 times larger, according to a study by a group of Portuguese scientists conducted in 2017. They ran a benchmark across 27 different programming languages to see if their energy consumption could be compared. The research showed that programming languages that are optimized for execution time rank better, as Table 1 shows.



Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Source: "Energy Efficiency across Programming Languages," Pereira et al.

>> Energy efficiency of programming languages (Table 1)

The study used multiple algorithms written in the programming languages listed in Table 1. It compared the amount of energy needed to run the code, the time it took to compute the algorithm, and the amount of data processed. The top three languages (C, C++, and Rust) are known to be designed and optimized heavily for their execution performance. The three best ranked languages for energy efficiency also rank best for execution time. This should not come as a surprise, as the formula to calculate the energy efficiency includes a variable for execution time as well as the amount of power needed for the execution. However, this does not prove time to be the biggest influence on energy efficiency. Looking at individual benchmarks in the table, the remaining list of programming languages shows cases with a very different order for both energy and time.

Actions You Can Take as a Developer

The programming languages used to build software can make a huge difference in the amount of energy websites and applications consume. When building a website, you can actively take measures to reduce its climate impact. Developers need to work together to find different ways to reduce the bundle size of a website and the bandwidth required to load its content. Choosing an energy efficient programming language can also make an impact on the energy consumption.

With an existing website or application, there is no need to rebuild it in a different programming language if you have used an energy inefficient language. One of the things developers can do is implement caching for the website. It can be enabled on both the website's frontend and backend parts, including the browser itself. Caching is a way to improve the loading speed by storing certain elements in memory, instead of on a server, and loading them from memory instead of reloading the information from the server every time somebody loads it. If you used a framework to create your website, you can often install a plugin to enable caching.

Aside from caching your website on a server, you can also use the browser for caching. With that, the visitor's browser will store (pieces of) the JavaScript bundle, fonts, stylesheets, and all the other data and content for the website. This not only speeds up the website's performance, but also prevents the browser from loading these files each time the browser visits the website, which in turn reduces the amount of requests to and from data centers. Most modern browsers automatically cache the contents of your website, and you can even instruct the browser how long to cache them for.

If you have used a JavaScript framework for building your website, you can reduce your bundle size by making use of a static website. This provides a friendlier solution for building a performant website with JavaScript. Static websites do not execute the JavaScript code in the browser when rendering the page, but rather beforehand. The visitor is served a pure HTML file that contains the complete web page and its content. This way, the JavaScript code is only executed when compiling the JavaScript bundle instead of during every visit. For most popular JavaScript frameworks, there are static website generators available. Examples include Gatsby, Next.js, and Hugo. They also provide additional functionality like PWA (progressive web app) support, automatic code splitting, and support for serverless functions.

Another possible optimization developers can implement to reduce the climate impact of their website is using a Content Delivery Network (CDN). CDNs act as a middleman between the website and its visitors. Instead of loading all image or video content from the site, the traffic runs through the CDN. CDNs help optimize the loading of images and videos by serving lower resolution content for visitors who use a device that does not support a higher resolution. A CDN also enables developers to serve multiple versions of their content, depending on other factors like the visitor's location, to reduce latency.

Switching to Green, Renewable Energy

Aside from all the possible improvements to websites and applications, the hosting location of the software greatly impacts the climate footprint. Switching the website’s hosting does not require any changes to the code. Therefore, it might be easier to switch the hosting provider than to change programming languages. Nowadays, the number of hosting providers to pick from is endless. However, choosing a web hosting company is not a decision to be taken lightly. Data centers in Europe are supposed to be climate neutral by 2030, and there is a way to check whether a data center is green.

The website of the Green Web Foundation lists all green data centers worldwide, so anyone can check whether their website is being hosted by a green data center. A green data center not only uses renewable energy to power its servers, but also uses energy that is purchased 100 percent carbon-free. In addition to this, it may also create reusable energy, in the same way that cities and factories have excess heat and power available for other houses or buildings nearby. We know that servers generate a lot of heat, so why not use that heat for other purposes? It can be used to heat cities.

Sustainability on Every Level

With technology in general improving at a rapid pace, server technology is improving as well. Servers are often split up into virtual and cloud servers, which are becoming more popular every year. As server technology improves, it means that we can host more and more websites on a single physical server. Thereby, businesses are able to cut costs, improve efficiency and also achieve more in terms of sustainability. Data centers cannot replace their servers every year, though, and if they do, that raises the question what will happen to the old servers. Reducing the carbon footprint of a website is therefore not only about its energy consumption, but about its sustainability on every level.

To summarize, there are multiple ways you as a developer can help fight climate change by being aware of, and acting on, the previously discussed aspects: the impact of large JavaScript bundle sizes and ways to make these smaller, the energy consumption of different programming languages, and the usage of green energy by data centers. The European Union wants to be climate-neutral by 2050, so the time to act is now. (mai)

Sources

All sources for this article can be found here: ix.de/zpuu

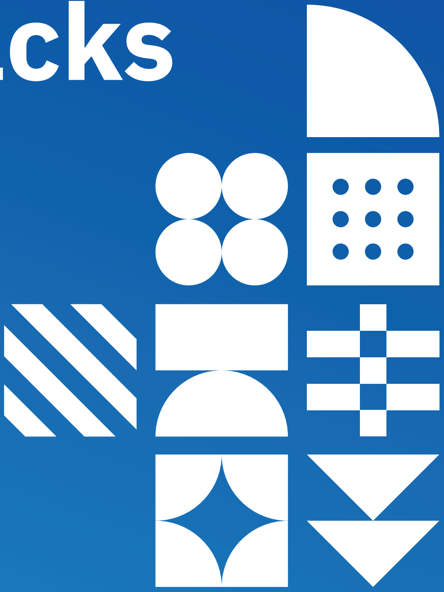
Roy Derks

is an entrepreneur, speaker and author from the Netherlands and, in his own words, "wants to make the world a better place through tech." He has been giving talks and trainings to developers worldwide on technologies like GraphQL, React and TypeScript. Most recently he wrote the book "Fullstack GraphQL."

Adversarial Attacks and Defenses

Christoph Reinders and Bodo Rosenhahn

Artificial neural networks have been showing remarkable successes, and machine learning is becoming an everyday companion even in critical areas. It is vital to protect applications that use machine learning from attacks.



Many machine learning (ML) applications are safety-critical, such as automated inspection in industrial production, support for physicians in evaluating computed tomography (CT) scans, traffic sign recognition in driver assistance systems, and road user detection in autonomous driving. All require high accuracy, stability, and reliability. The consequences of not, or incorrectly, recognizing a stop sign, for example, can be devastating. Therefore, the analysis of the robustness and vulnerability of neural networks is of particular importance.

In a Nutshell

- Machine learning applications have become ubiquitous and are present in safety-critical areas such as driver assistance systems in cars.
- Neural networks possess certain vulnerabilities and may misread images, including traffic signs, that have been manipulated in a way that is imperceptible to the human eye.
- There are several options to harden neural networks against these adversarial attacks, for example adversarial training.

In recent years, several attacks have demonstrated the vulnerability of neural networks. Simple and hardly noticeable manipulations of (image) data may lead to networks predicting completely wrong results. The neural networks may even attest high confidence for the results, meaning that they are quite certain for the wrong result to be correct. Recent attacks, such as the one on Tesla vehicles [1], have shown that the techniques are not just theoretical but also have a definite impact in the real world.

Targeted Manipulation

Vulnerabilities in these algorithms raise some questions, such as which patterns enable an attack on neural networks. In the future, when people are choosing an outfit, will they have to worry about whether their T-shirt might include a pattern that is not recognized by driver assistance systems or may confuse autonomous vehicles?

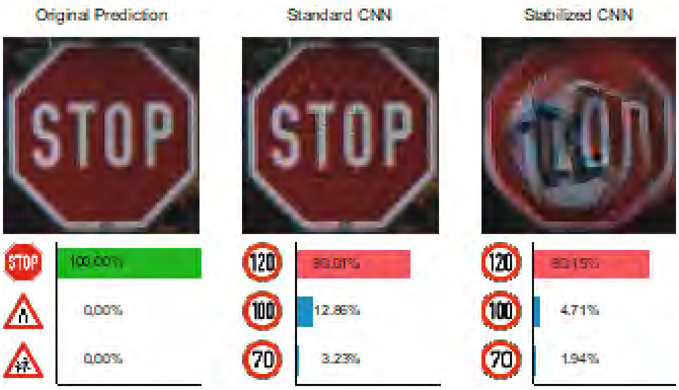
Attackers can create manipulated images that differ only slightly from normal images, but are intentionally altered to mislead a computer model into making mistakes. For the human eye, the changes are often invisible or can only be detected by looking very closely at the image. Many of these methods are based on the calculation of gradients. As with backpropagation during the training of neural networks, a target function is optimized, and the gradients are propagated backwards from the output to the inputs of the network—in this case, all the way back to the pixel values. For each pixel, the methods calculate how the pixel value would have to be changed to trigger an erroneous decision. Iteratively, subtle manipulations are applied to the image until the neural network is fooled by the new image. The manipulated image is not comprised of random noise or arbitrary patterns, even if it may seem so. The patterns are optimized to attack neural networks with subtle changes and to trigger wrong decisions.

An example is shown in Figure 1. The classifier correctly recognizes the original image as a stop sign. After a targeted manipulation, the neural network sees a speed limit of 120 km/h.

Real-World Attacks

These kind of attacks are based on two crucial assumptions: The attacker has direct access to the input data of the artificial intelligence (AI) system and also possesses all information about the neural network, such as its architecture and learned weights. In reality, the attacks are much more complex. By accessing a system, control commands can be manipulated directly during the attack. Otherwise, the manipulation of the objects or the environment must be done in the real world, and the AI system captures the input data via a camera. What makes it even more difficult is that the parameters of neural networks are often unknown in proprietary applications.

Several methods have shown that neural networks can be manipulated in the real world. Researchers from the Belgian university KU Leuven have developed a method to fool person recognition systems [2]. A generated and printed pattern makes the person carrying it undetectable by the system (see Figure 2).



>> Subtle manipulations are sufficient to fool a neural network. A stop sign, for example, is classified with high confidence as a speed limit of 120 km/h because barely perceptible patterns have been added to the traffic sign. Adversarial training is one way to make neural networks more robust (Figure 1).

While the pattern is obvious to observers, there are inconspicuous techniques to manipulate the recognition of traffic signs [3]. Harmless-looking changes made by stickers or graffiti on traffic signs may cause a machine learning system to either miss a traffic sign or to recognize it wrongly. Tesla's Driver-assistance systems have already become targets of attacks. Inconspicuous-looking white dots on the road [1], or attacks hidden in advertisements [4], can also cause the ML system to make incorrect decisions.

Adversarial Training

Various strategies have been developed to protect neural networks from attacks. One possible option is adversarial training, which incorporates manipulated images in the training process to increase the robustness of the neural network. The goal is for the cost of attacks to be as high as possible, and to require manipulations that are clearly recognizable for humans.

In adversarial training, as in regular training of neural networks, the algorithm is fed with samples, and the weights for these are gradually adjusted. Now, however, some samples are replaced with manipulated images that the system continuously regenerates during the training process. This way, the current model of the network is constantly analyzed. By incorporating the manipulated training data, the neural network learns from its own vulnerabilities and becomes more robust.

The effect of adversarial training is shown in Figure 3. The target class is displayed in the first column and the original image in the second column. From the third column on, manipulated images are shown in the respective training epochs. The manipulated images are generated with the goal that the confidence of the target class is above 80 percent.

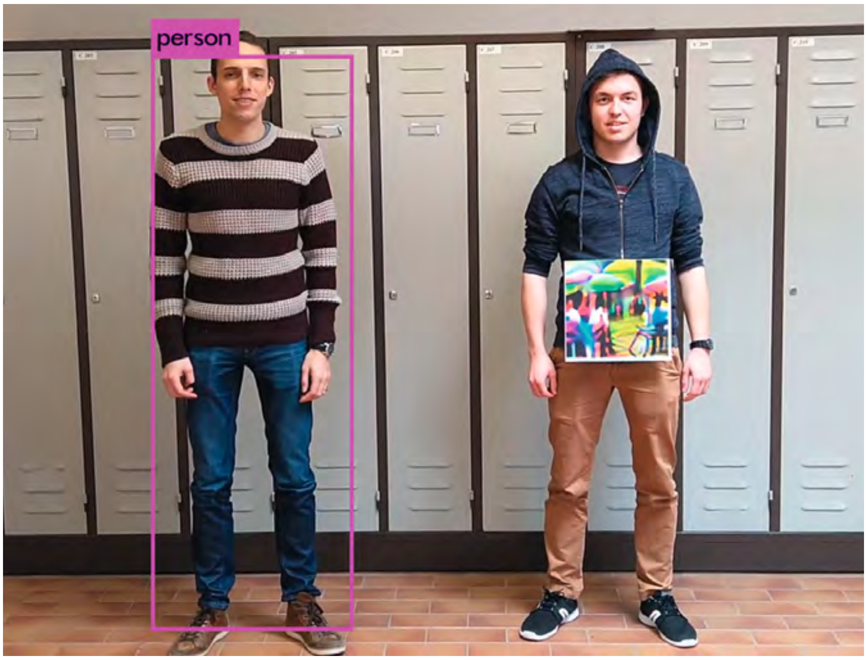
In the beginning, the neural network is easily attackable with manipulated images. The changes are largely imperceptible to humans. During the training process, the effort to generate manipulated images increases significantly. The analysis of the progress shows that the examples in the last epoch are transformed into the target class (first and second row), or that visual features of the target class are added (third and fourth row). Overall, the attacks require major image manipulations that are more noticeable to the human eye.

There are additional defense methods: Autoencoders can be used to preprocess the manipulated images. Since the manipulation often occurs in the high-frequency components, the images can be smoothed or denoised by an autoencoder. Another possibility is gradient masking: Many methods are based on the calculation of gradients, and this defense approach follows the strategy of intentionally hiding the corresponding information of the model to confuse attackers. A final example is the detection of manipulated images: Classifiers trained on normal and manipulated images can detect if the system is under attack.

As in many scenarios, this is a cat-and-mouse game: New attack methods are followed by methods of defending the attacks, which in turn are followed by more sophisticated attacks. Researchers are constantly developing new methods for attacking and defending neural networks, such as generating transferable perturbations [5], self-supervised representation learning approaches to purify adversarial examples [6], or attacks in new domains, like semantic segmentation.

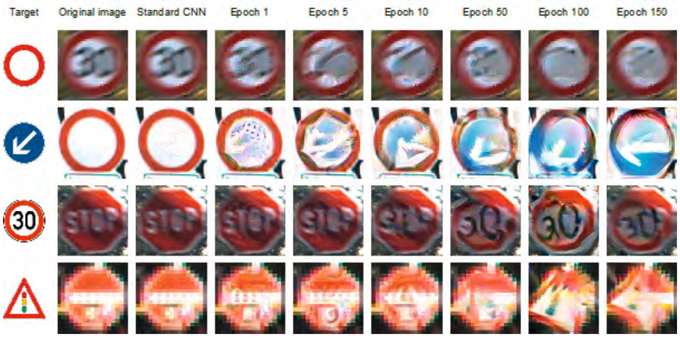
Challenging Defense

Defending these systems is not simple, due to their complexity. Neural networks are trained on datasets with thousands or even millions of images. When looking at the number of all possible inputs, training datasets cover only a tiny fraction of the possibilities. In the high-dimensional feature space, small changes to each pixel can have a significant impact during propagation through the neural network. As a result, it is easy to generate examples that the neural network has never seen before and which are not covered by the training data.



Source: Conference paper at CVPRW, Thys et al. [2]

>> This pattern was specifically optimized to fool a person recognition system. Printed out, it makes the person with the pattern invisible to the machine learning system in question (Figure 2).



>> Adversarial training analyzes the vulnerabilities and hardens a neural network so that significantly more manipulation is necessary during the training process and the attack becomes noticeable to observers (Figure 3).

The demands on a neural network are generally high. For every input, it should provide a suitable answer. Alternatively, the neural network should return a confidence score indicating the certainty with which it has arrived at its decision. For an example that is outside of the training data distribution, the neural network should predict low confidence.

Conclusion

Although artificial neural networks are constantly achieving new heights in various applications, such as classification, object recognition, or semantic segmentation, they can be fooled by subtle manipulations that are barely visible to humans. This shows that neural networks still have a lot to learn—especially with the way humans learn as role model.

Attack and defense tactics will evolve in the coming years. Many attacks require knowledge of the model parameters, possession of the training data, direct access to the AI system to manipulate the input data, or are optimized for specific architectures. This makes them complex and difficult to achieve in the real world.

As a rule of thumb, don't trust an AI that you haven't trained yourself! As with running third-party code, developers must verify the source, validate the training process, and pay attention to quality and diversity of the training data. Tools such as Foolbox or the Adversarial Robustness Toolbox (all links for this article: ix.de/z3kg) implement several methods for examining a neural network for vulnerabilities. To provide a knowledge database, Mitra and other companies like Microsoft, Nvidia, Bosch, and IBM have published the Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS).

Sources

All links for this article can be found here: ix.de/z3kg

[1] Tencent Keen Security Lab, "Experimental Security Research of Tesla Autopilot." https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf, 2019.

[2] Thys, Simen et al. "Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[3] Eykholt, Kevin et al. “Robust Physical-World Attacks on Deep Learning Visual Classification.” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] Nassi, Ben et al. “Phantom of the ADAS: Phantom Attacks on Driver-Assistance Systems.“ *Cryptology ePrint Archive*, 2020.

[5] Naseer, Muzammal et al. “On Generating Transferable Targeted Perturbations.” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[6] Shi, Changhao et al. “Online Adversarial Purification based on Self-supervised Learning.” *International Conference on Learning Representations (ICLR)*, 2021.

[7] Nesti, Federico et al. “Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks.“ *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

(mai)

Christoph Reinders

is a research associate at the Institute for Information Processing at Leibniz University Hannover. His research interests include image processing and machine learning.

Bodo Rosenhahn

is a professor at the Institute for Information Processing at Leibniz University Hannover. He studied computer science with a minor in medicine at Kiel University. He has published more than 180 articles, book chapters and scientific papers. In his work, he has been focusing on video and image analysis as well as deep learning.

Contact

iX – Magazin für professionelle IT
Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10,
30625 Hannover, Germany
Phone: 0511 5352-387, fax: 0511 5352-361, **e-mail:** post@ix.de

Publishers: Ansgar Heise, Christian Heise

Editor-in-Chief: Dr. Oliver Diedrich (responsible for the text section)

Concept and Editorial Leadership: Maika Möbus (mai@ix.de) -589

Editor: Rainald Menge-Sonnentag (rme@ix.de) -884

Authors of this Issue:
Stefan Baumgartner, Adam Bien, Stefania Chaplin, Roy Derks, Hendrik Ebbers, Sandra Gerberding, Ivar Grimstad, Miško Hevery, Maika Möbus, Tobias Modig, Sandra Parsick, Christoph Reinders, Bodo Rosenhahn, Falk Sippach, Daniel Strmečki, Wolfgang Weigend

DTP Production:
Lisa Hemmerling, Heise Medienwerk, Rostock

Proofreading and Final Correction:
Fabian A. Scherschel

Cover Image:
© Freepik; Montage: Lisa Hemmerling

Publishing House:
Heise Medien GmbH & Co. KG,
Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover, Germany;
phone: 0511 5352-0, fax: 0511 5352-129

Managing Directors:
Ansgar Heise, Beate Gerold

Members of the Executive Board:
Jörg Mühle, Falko Ossmann

Advertising Director (responsible for the advertising section):
Michael Hanke (-167),
e-mail: michael.hanke@heise.de,
www.heise.de/mediadaten/ix

Head of Sales and Marketing:
André Lux (-299)

Printed by:
Dierichs Druck + Media GmbH & Co. KG,
Frankfurter Straße 168, 34121 Kassel, Germany

Despite thorough review by the editorial team, the publishing house cannot accept any liability for the correctness of the publications. No part of this publication may be distributed without the explicit written permission of the publishing house; this explicitly includes publication on websites.

Printed in Germany

Advertisers

AIT Austrian Institute of Technology GmbH	A-Wien	27	InterSystems GmbH	Darmstadt	9
AKKA Deutschland GmbH	Sindelfingen	29	NWB Verlag GmbH & Co. KG	Herne	35
anynines GmBH	Saarbrücken	33	OPTIMAL SYSTEMS GmbH	Berlin	25
attempto GmbH & Co. KG	Aschheim	41	OutSystems	Frankfurt	21
Bertrandt AG	Ehningen	32	Robert Bosch AG	A-Wien	13
Bittner+Krull Softwaresysteme GmbH	München	39	Rohlik Group a.s.	CZ-Karlin	2
Blokhaus Inc	USA-Newark	7	Rosen Technology and Research Center GmbH	Lingen	11
IABG Industrieanlagen-Betriebsgesellschaft mbH	Ottobrunn	19	TOPdesk Deutschland GmbH	Kaiserslautern	23
IBM iX Berlin GmbH	Berlin	5	XITASO GmbH IT & Software Solutions	Augsburg	31
Infobip Ltd	GB-London	44			

The page numbers printed here are not binding. Editorial reasons may make changes necessary.

betterCode()   building IoT   c't <webdev>   Continuous Lifecycle   [Container Conf]

data2day   enterPy   // heise devSec()   heise MacDev   Herbstcampus

enterJS   inside agile   Industrial IoT CONFERENCE   Minds Mastering Machines   storage2day

ONLINE & IN PERSON

DEVELOPER June – Dec 2022 CONFERENCES

For further information please visit:
heise.de/developer

Organisers:

iX   heise Developer   dpunkt.verlag

- **22 – 23 June, 2022**
  enterJS (Darmstadt)
  enterjs.de
- **23 June, 2022**
  Industrial IoT Conference (Munich)
  iiot-conference.de
- **29 June – 2 July, 2022**
  CloudLand (Phantasialand Brühl)
  cloudland.org
- **30 June, 2022**
  4th Product Owner Day (POD) (online)
  pod.inside-agile.de
- **6 – 8 September, 2022**
  Herbstcampus (Nuremberg)
  herbstcampus.de
- **20 – 21 September, 2022**
  data2day (Karlsruhe)
  data2day.de
- **5 – 6 October, 2022**
  heise devSec (Karlsruhe)
  heise-devsec.de
- **13 – 14 October, 2022**
  storage2day (Munich)
  storage2day.de
- **25 October, 2022**
  enterPy (online)
  enterpy.de
- **26 October, 2022**
  Team Up! (online)
  teams.inside-agile.de
- **15 – 18 November, 2022**
  Continuous Lifecycle and
  ContainerConf (Mannheim)
  continuouslifecycle.de
- **29 November, 2022**
  Agile Leadership Conference (ALC) (online)
  alc.inside-agile.de
- **6 December, 2022**
  5th Product Owner Day (POD) (online)
  pod.inside-agile.de