Securing Angular with Trusted Types

Dr. Philippe De Ryck

https://Pragmatic Web Security.com

```
Philippe De Ryck<img src="none" onerror="alert('OMG')">
```

```
<h1>
  Welcome Philippe De Ryck
  <img src="none"
  onerror="alert('OMG')">
</h1>
```

Multiple XSS vulnerabilities in child monitoring app Canopy 'could risk location leak'

Jessica Haworth 06 October 2021 at 14:25 UTC
Updated: 07 October 2021 at 09:09 UTC

XSS Vulnerabilities Mobile

Pair of unpatched security bugs are 'just the tip of the iceberg'

Facebook pays out $25k bug bounty for chained DOM-based XSS

Adam Bannister 09 November 2020 at 17:55 UTC
Updated: 11 November 2020 at 11:45 UTC

Bug Bounty | Social Media | XSS

Researcher awarded five-figure sum for 'easy to exploit' bug

Oculus, Facebook account takeovers net security researcher $30,000 bug bounty

Adam Bannister 05 January 2021 at 15:28 UTC
Updated: 06 January 2021 at 11:16 UTC

Facebook Bug Bounty XSS

XSS in virtual reality forum one of three flaws chained to land bumper payout

https://portswigger.net/daily-swig/oculus-facebook-account-takeovers-net-security-researcher-30-000-bug-bounty

Trusted Types has the ability to eradicate DOM-based XSS in your entire application

I am *Dr. Philippe De Ryck*

Founder of Pragmatic Web Security

Google Developer Expert

Auth0 Ambassador

SecAppDev organizer

I help developers with security

✓ Hands-on in-depth security training

✓ Advanced online security courses

✓ Security advisory services

https://pragmaticwebsecurity.com

```
Philippe De Ryck<img
    src="none"
onerror="alert('OMG')">
```

An Angular template to put data into the page

```
<h1> Welcome {{ name }} </h1>
```

Angular applies automatic escaping to data embedded in templates

The data seen by the browser

```
Philippe De Ryck&lt;img src="none"
onerror="alert('OMG')"&gt;
```

The browser does not see HTML code, but simply renders the HTML tags

ee greatest things you learn from traveling

the great things on earth traveling teaches us by example. Here are some of the mo ous lessons I've learned over the years of traveling.

Leaving your comfort zone might lead you to such beautiful sceneries like this one.

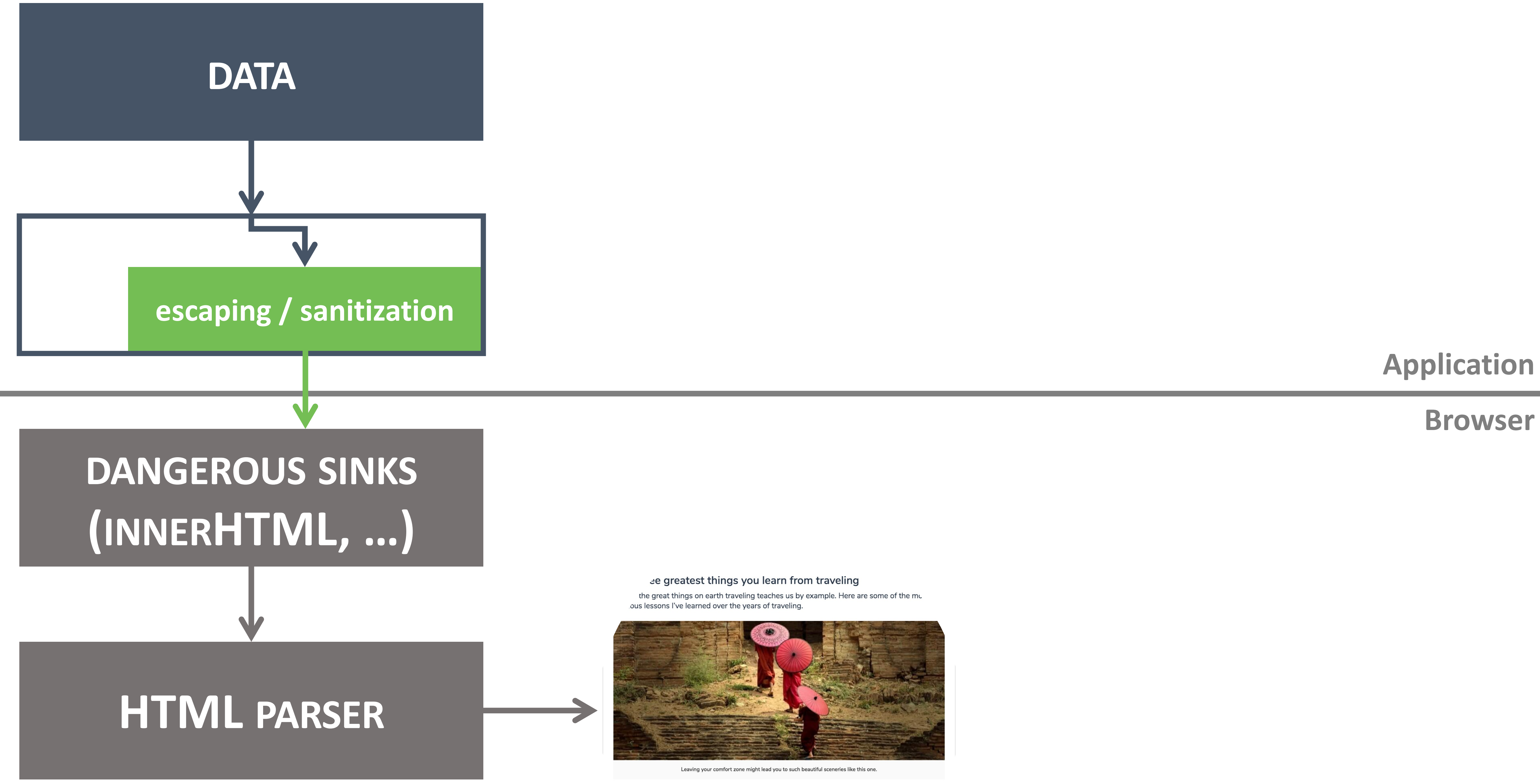
ppreciation of diversity

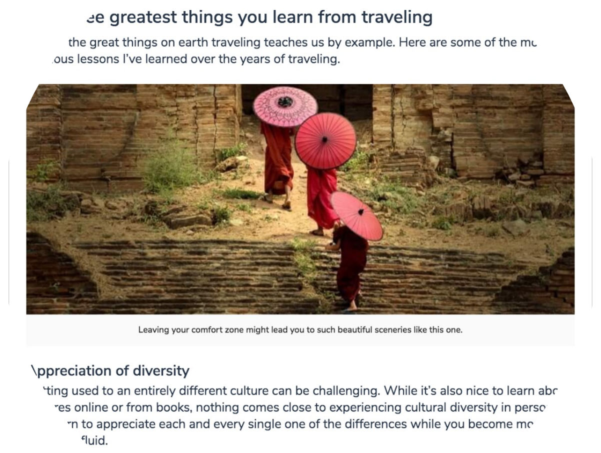
tting used to an entirely different culture can be challenging. While it's also nice to learn abo res online or from books, nothing comes close to experiencing cultural diversity in perso rn to appreciate each and every single one of the differences while you become mo fluid.

An Angular template to render user-provided HTML

```
<div>
  <h3>{{ review.title }}</h3>
  <p [innerHTML]="review.content"></p>
</div>
```

[innerHTML] **does not directly expose *innerHTML* property, but sanitizes the data first**




꜀e greatest things you learn from traveling

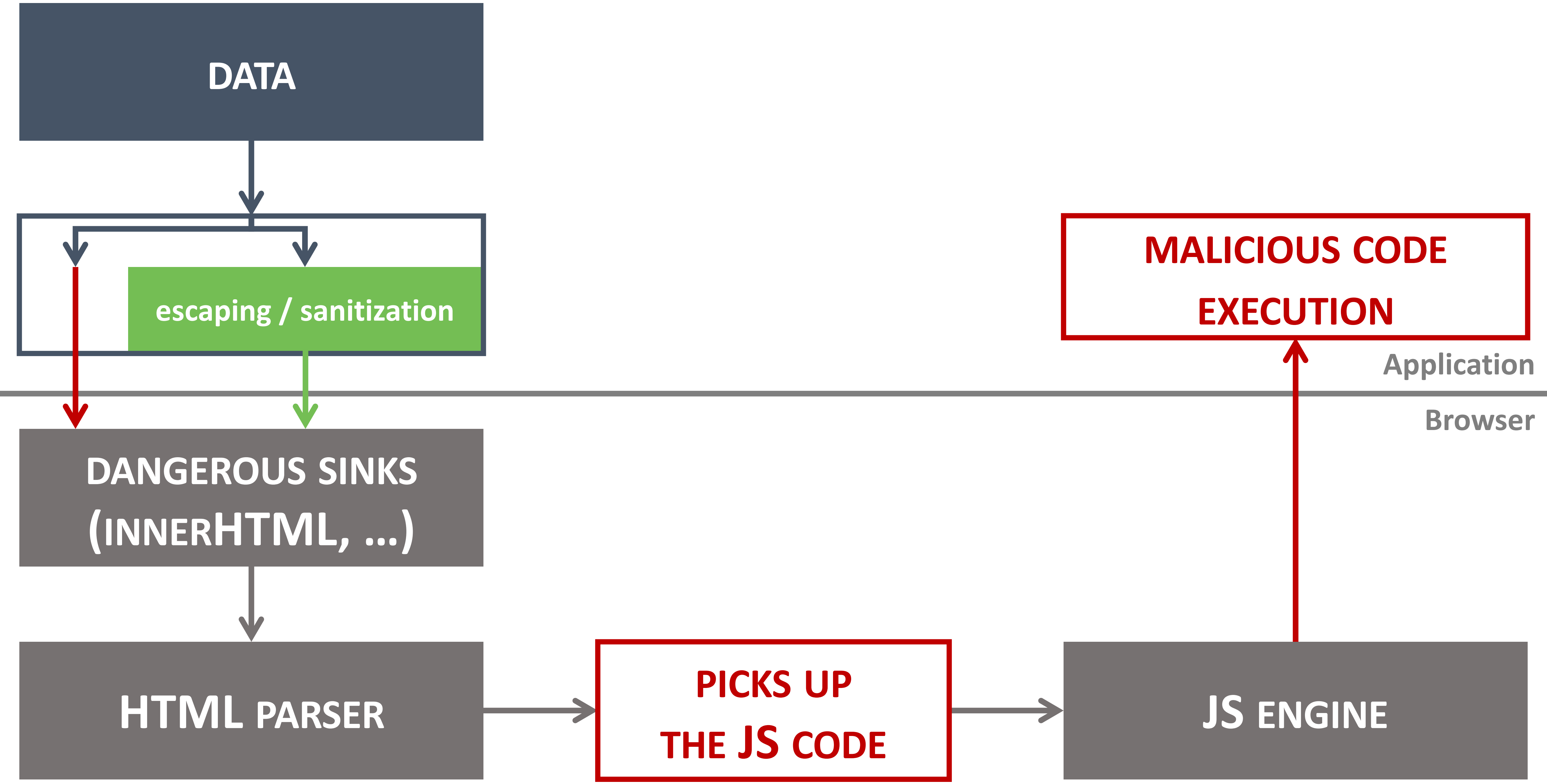
the great things on earth traveling teaches us by example. Here are some of the mu
ous lessons I've learned over the years of traveling.

Leaving your comfort zone might lead you to such beautiful sceneries like this one.

\ppreciation of diversity

ʻting used to an entirely different culture can be challenging. While it's also nice to learn abr
ʻes online or from books, nothing comes close to experiencing cultural diversity in perso
ʻn to appreciate each and every single one of the differences while you become mo
ʻluid.

@PhilippeDeRyck



@PhilippeDeRyck

bypassSecurityTrustHTML

An example using bypassSecurityTrustHTML

```
let unsafeValue = this.sanitizer.bypassSecurityTrustHtml(review);
```

Assigning this value to *[innerHTML]* causes an XSS vulnerability

Angular code to obtain a native DOM element

```
@ViewChild("myDiv") div : ElementRef;
```

Angular code to obtain a native DOM element

```
@ViewChild("myDiv") div : ElementRef;
```

With Angular out of the loop, bad things are bound to happen

```
this.div.nativeElement.innerHTML = this.inputValue;
```

With *ElementRef*, you can access native DOM elements, where Angular cannot apply automatic protection against XSS

Angular code to obtain a native DOM element

```
@ViewChild("myDiv") div : ElementRef;
```

With Angular out of the loop, bad things are bound to happen

```
this.div.nativeElement.innerHTML = this.inputValue;
```

The Angular Renderer2 API also allows native access to the DOM

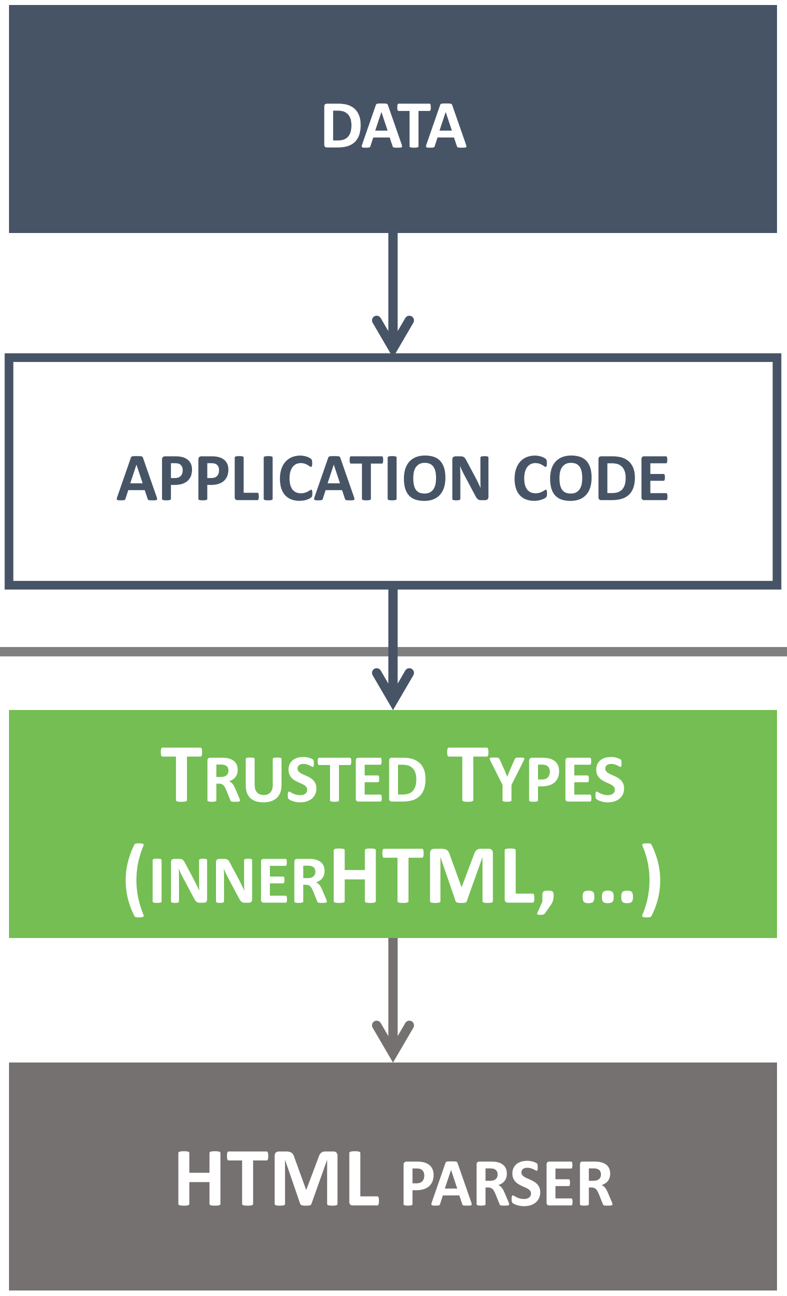
```
constructor(private renderer2 : Renderer2) {}

loadDivWithRenderer2() {
  this.renderer2.setProperty(this.div, "innerHTML", this.inputValue);
}
```

With *Renderer2*, you can access native DOM elements, where Angular does not apply automatic protection against XSS

XSS in Angular applications is not impossible

Angular offers secure-by-default mechanisms, but a single mistake can still result in XSS vulnerabilities

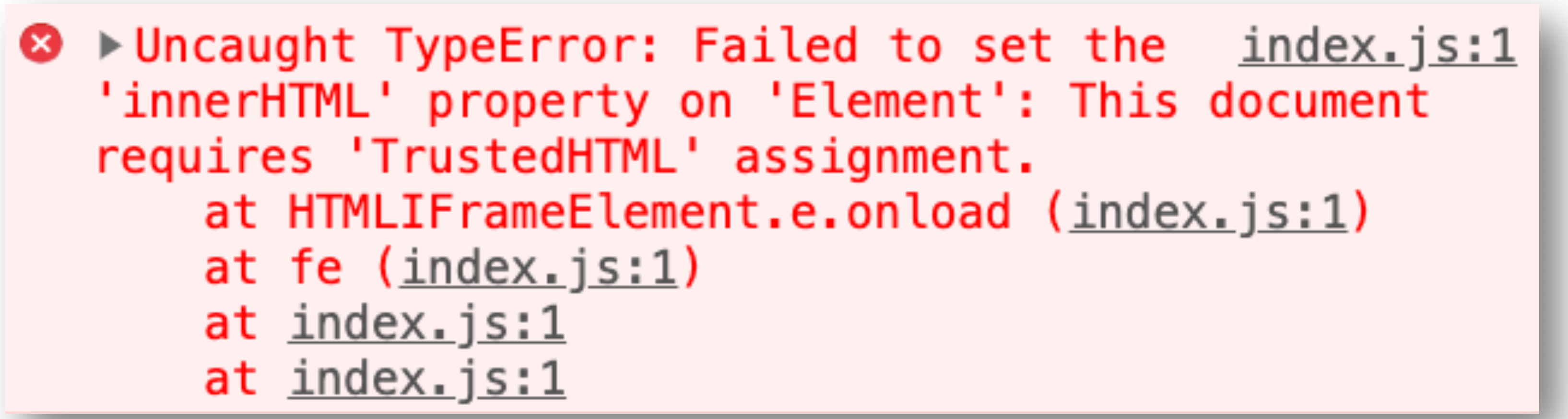


Enable trusted types by setting a CSP policy

```
Content-Security-Policy:
  require-trusted-types-for 'script'
```

Application

Browser



```
▸Uncaught TypeError: Failed to set the    index.js:1
'innerHTML' property on 'Element': This document
requires 'TrustedHTML' assignment.
    at HTMLIFrameElement.e.onload (index.js:1)
    at fe (index.js:1)
    at index.js:1
    at index.js:1
```

@PhilippeDeRyck

Trusted Types complement static analysis by providing runtime guarantees about the absence of uncontrolled data flows in client-side code. Our analysis of the vulnerabilities reported to Google VRP shows that Trusted Types could **effectively prevent at least 61% of DOM XSS-es** missed by our static analysis pipeline.

Enable trusted types by setting a CSP policy

```
Content-Security-Policy: require-trusted-types-for 'script'
```

Tells the browser to only allow trusted types in the DOM

Trusted Types does not affect the use of proper DOM APIs

```
let msg = document.createElement("span");
msg.setAttribute("class", "italic");
msg.textContent = e.data;
document.getElementById("msg").appendChild(msg);
```

When possible, always opt to write clean code instead of relying on the browser's HTML parser

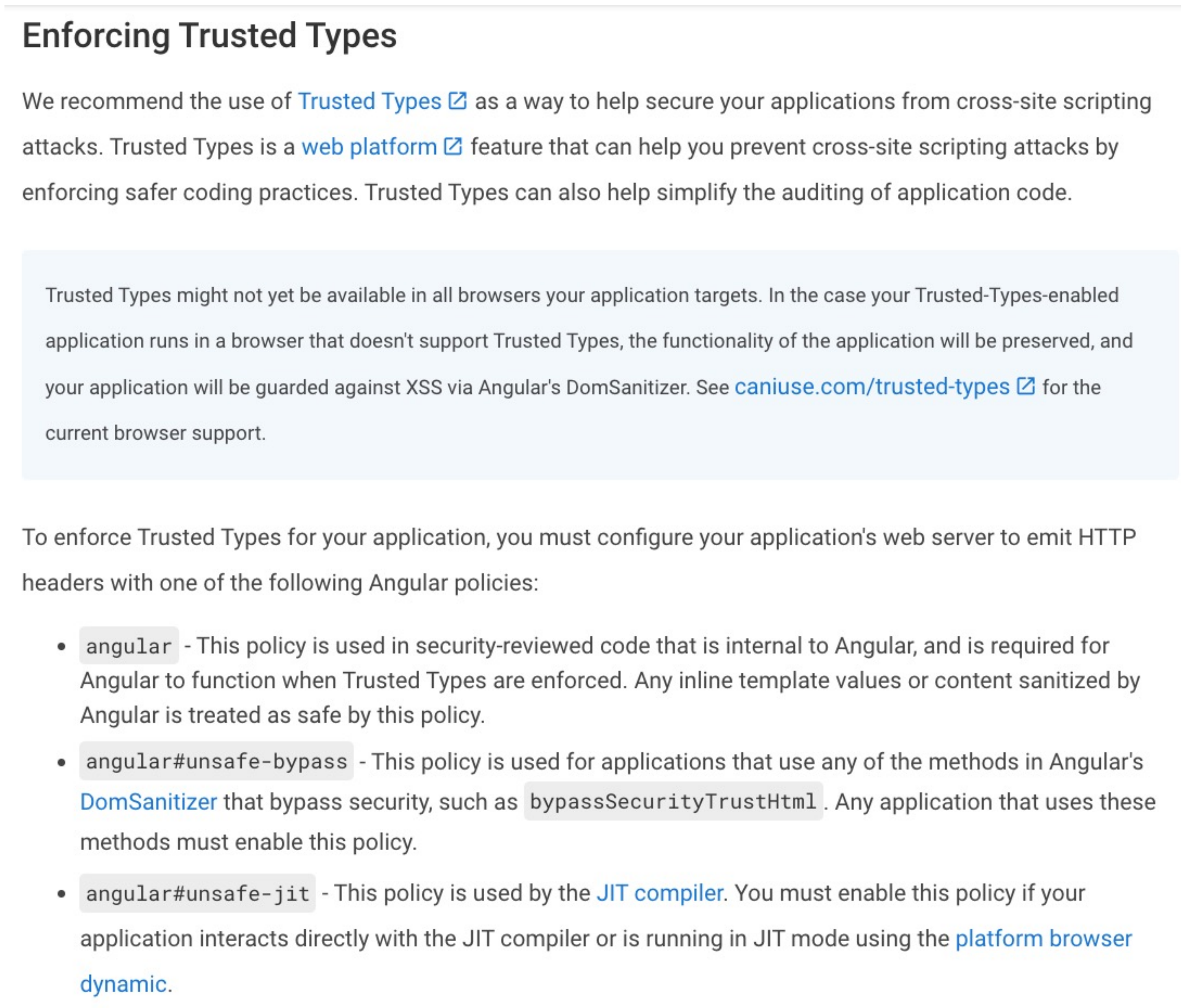
ee greatest things you learn from traveling

the great things on earth traveling teaches us by example. Here are some of the mo
ous lessons I've learned over the years of traveling.

Leaving your comfort zone might lead you to such beautiful sceneries like this one.

ppreciation of diversity

tting used to an entirely different culture can be challenging. While it's also nice to learn abo
res online or from books, nothing comes close to experiencing cultural diversity in perso
rn to appreciate each and every single one of the differences while you become mo
fluid.



Enforcing Trusted Types

We recommend the use of Trusted Types as a way to help secure your applications from cross-site scripting attacks. Trusted Types is a web platform feature that can help you prevent cross-site scripting attacks by enforcing safer coding practices. Trusted Types can also help simplify the auditing of application code.

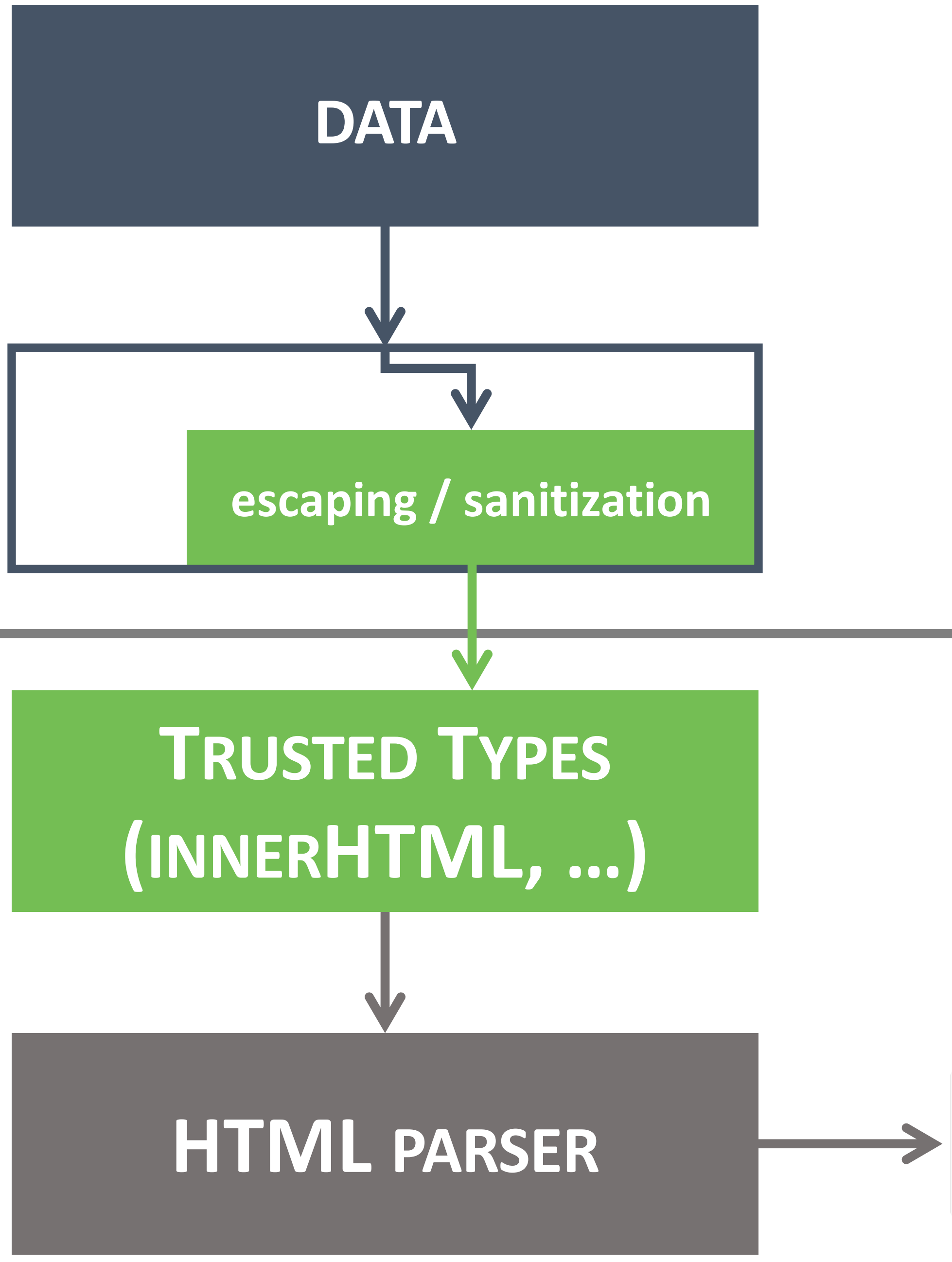
> Trusted Types might not yet be available in all browsers your application targets. In the case your Trusted-Types-enabled application runs in a browser that doesn't support Trusted Types, the functionality of the application will be preserved, and your application will be guarded against XSS via Angular's DomSanitizer. See caniuse.com/trusted-types for the current browser support.

To enforce Trusted Types for your application, you must configure your application's web server to emit HTTP headers with one of the following Angular policies:

- `angular` - This policy is used in security-reviewed code that is internal to Angular, and is required for Angular to function when Trusted Types are enforced. Any inline template values or content sanitized by Angular is treated as safe by this policy.
- `angular#unsafe-bypass` - This policy is used for applications that use any of the methods in Angular's DomSanitizer that bypass security, such as `bypassSecurityTrustHtml`. Any application that uses these methods must enable this policy.
- `angular#unsafe-jit` - This policy is used by the JIT compiler. You must enable this policy if your application interacts directly with the JIT compiler or is running in JIT mode using the platform browser dynamic.

@PhilippeDeRyck

https://angular.io/guide/security#enforcing-trusted-types

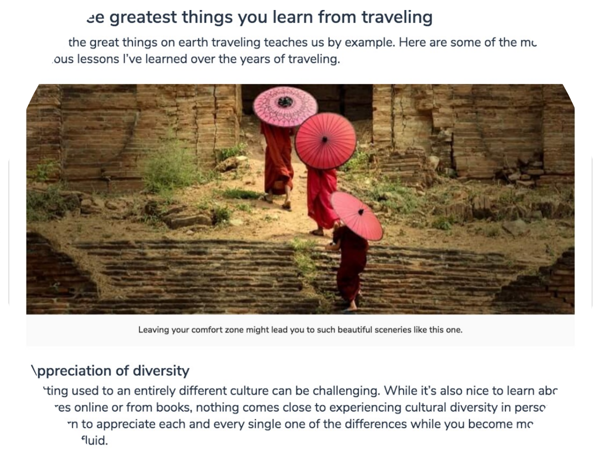


Enable trusted types for Angular applications

```
Content-Security-Policy:
  require-trusted-types-for 'script';
  trusted-types angular
```

Application

Browser


ɛe greatest things you learn from traveling

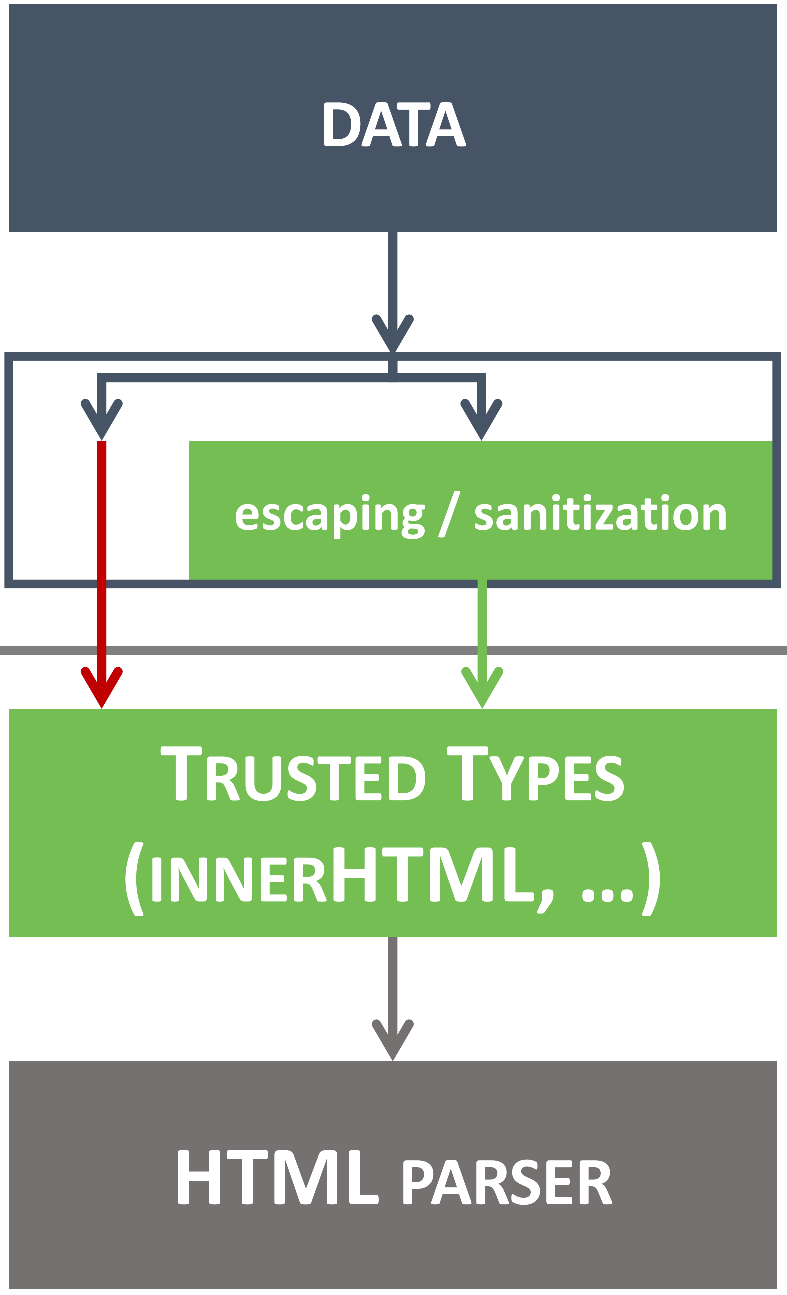
the great things on earth traveling teaches us by example. Here are some of the mc
ous lessons I've learned over the years of traveling.

Leaving your comfort zone might lead you to such beautiful sceneries like this one.

\ppreciation of diversity

ˑting used to an entirely different culture can be challenging. While it's also nice to learn abr
ˑes online or from books, nothing comes close to experiencing cultural diversity in persc
ˑn to appreciate each and every single one of the differences while you become mc
ˈluid.

@PhilippeDeRyck

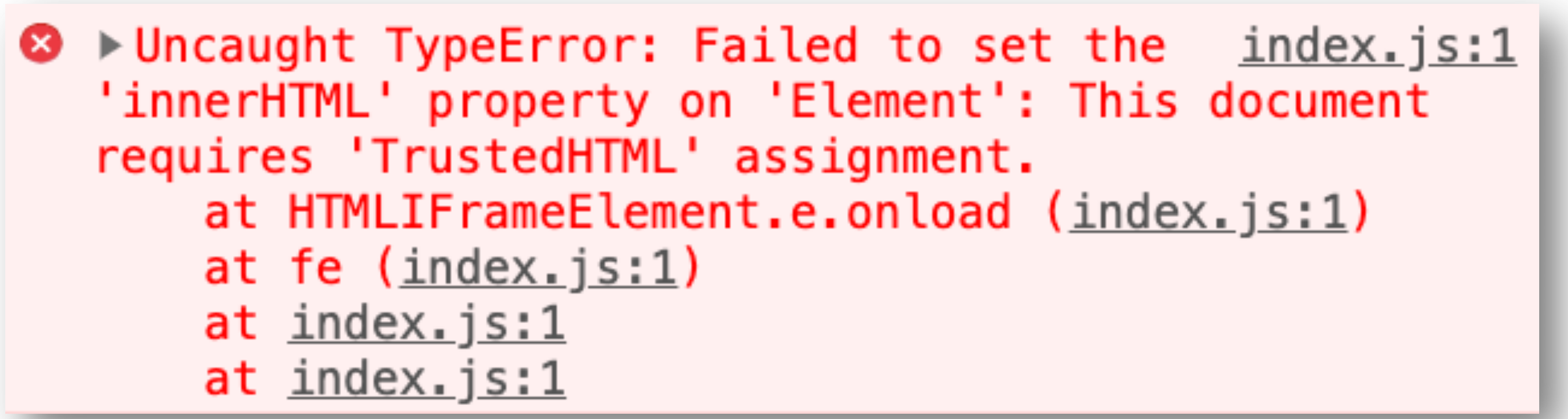


Enable trusted types for Angular applications

```
Content-Security-Policy:
  require-trusted-types-for 'script';
  trusted-types angular
```

Application

Browser



```
Uncaught TypeError: Failed to set the                    index.js:1
'innerHTML' property on 'Element': This document
requires 'TrustedHTML' assignment.
    at HTMLIFrameElement.e.onload (index.js:1)
    at fe (index.js:1)
    at index.js:1
    at index.js:1
```

@PhilippeDeRyck

TT forces you to transform text to a Trusted Type, it does not automatically apply security

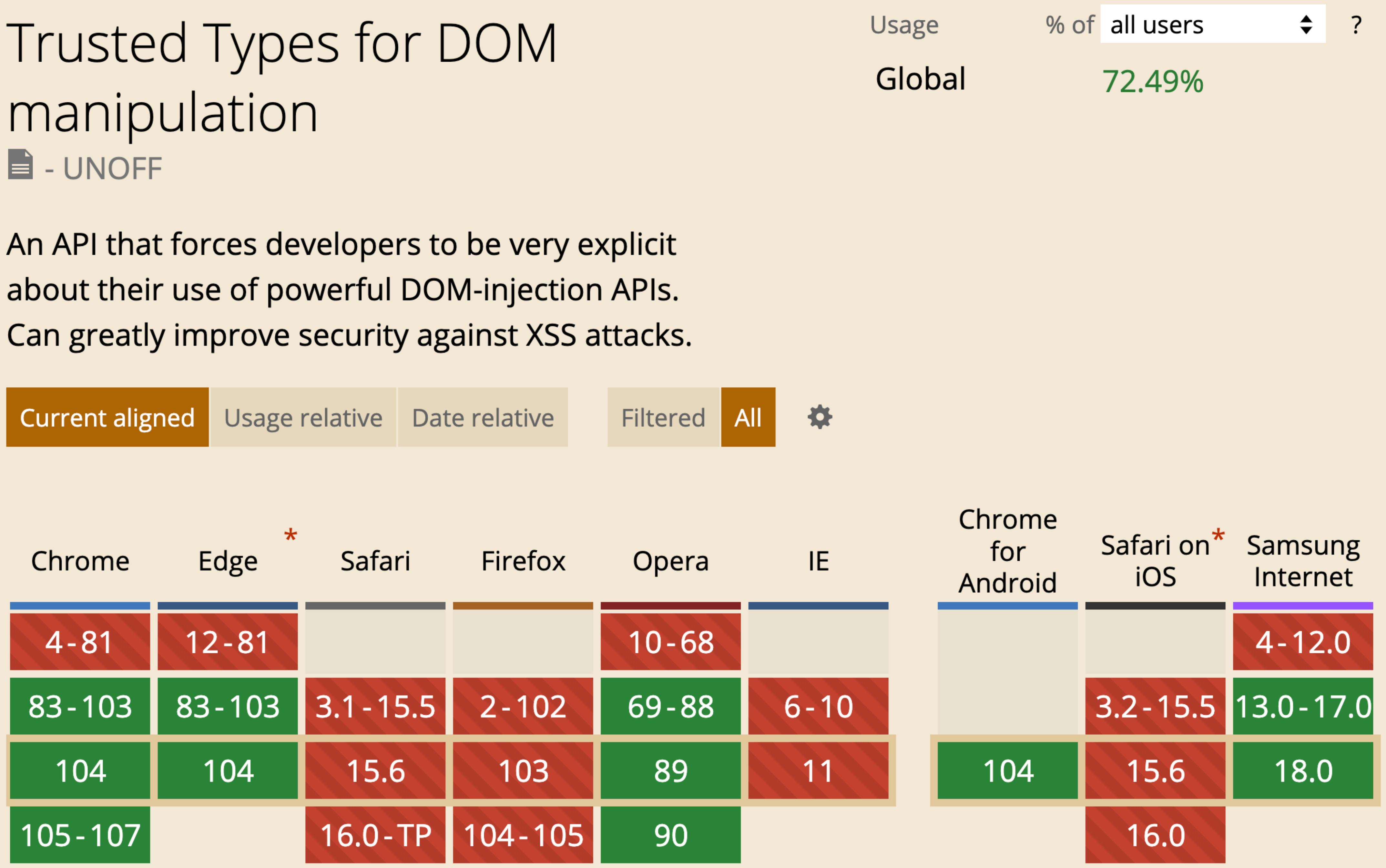
Weakening trusted types to allow bypassSecurityTrustHTML

```
Content-Security-Policy:
  require-trusted-types-for 'script';
  trusted-types angular angular#unsafe-bypass
```

This configuration allows the use of *bypassSecurityTrustHTML*, regardless of whether you use it securely or not

Trusted Types avoids unsafe DOM assignments

Enabling Trusted Types modifies default browser behavior, refusing the insecure usage of dangerous sinks in the DOM



Trusted Types for DOM manipulation

- UNOFF

Usage % of all users ?

Global 72.49%

An API that forces developers to be very explicit about their use of powerful DOM-injection APIs. Can greatly improve security against XSS attacks.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet
🟥 4-81	🟥 12-81			🟥 10-68				🟥 4-12.0
🟩 83-103	🟩 83-103	🟥 3.1-15.5	🟥 2-102	🟩 69-88	🟥 6-10		🟥 3.2-15.5	🟩 13.0-17.0
🟩 104	🟩 104	🟥 15.6	🟥 103	🟩 89	🟥 11	🟩 104	🟥 15.6	🟩 18.0
🟩 105-107		🟥 16.0-TP	🟥 104-105	🟩 90			🟥 16.0	

Source: caniuse.com

Enable trusted types by setting a CSP policy

```
Content-Security-Policy: require-trusted-types-for 'script'
```

With trusted types enabled, the browser refuses to assign text to innerHTML

```
this.div.nativeElement.innerHTML = this.inputValue;
```

Fixing the application for Chrome typically results in applying proper protections

```
<div [innerHTML]="inputValue"></div>
```

Enable trusted types by setting a CSP policy

```
~~Content-Security-Policy: require-trusted-types-for 'script'~~
```

Thanks to trusted types, the application follows security best practices

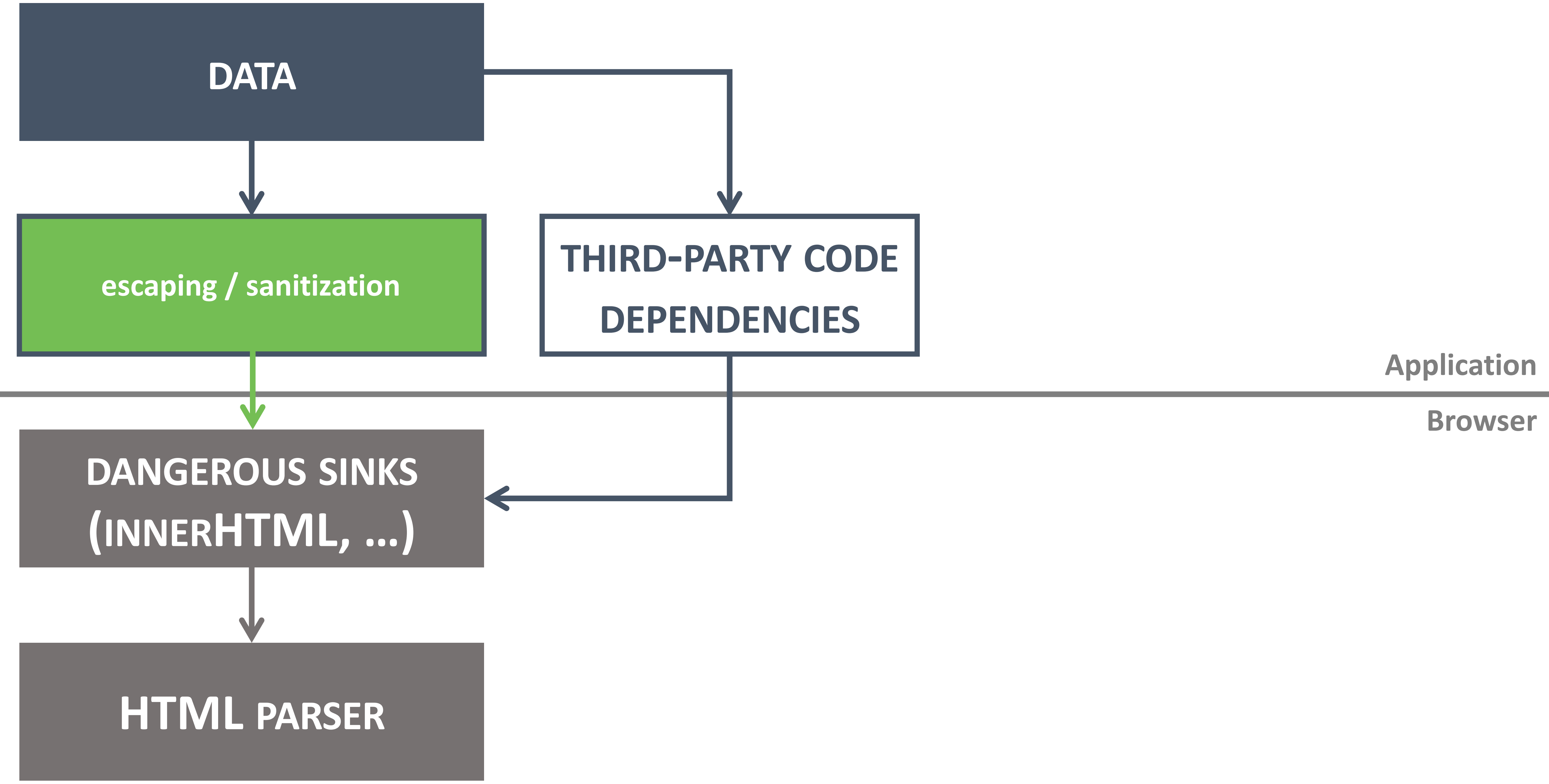
```
<div [innerHTML]="inputValue"></div>
```

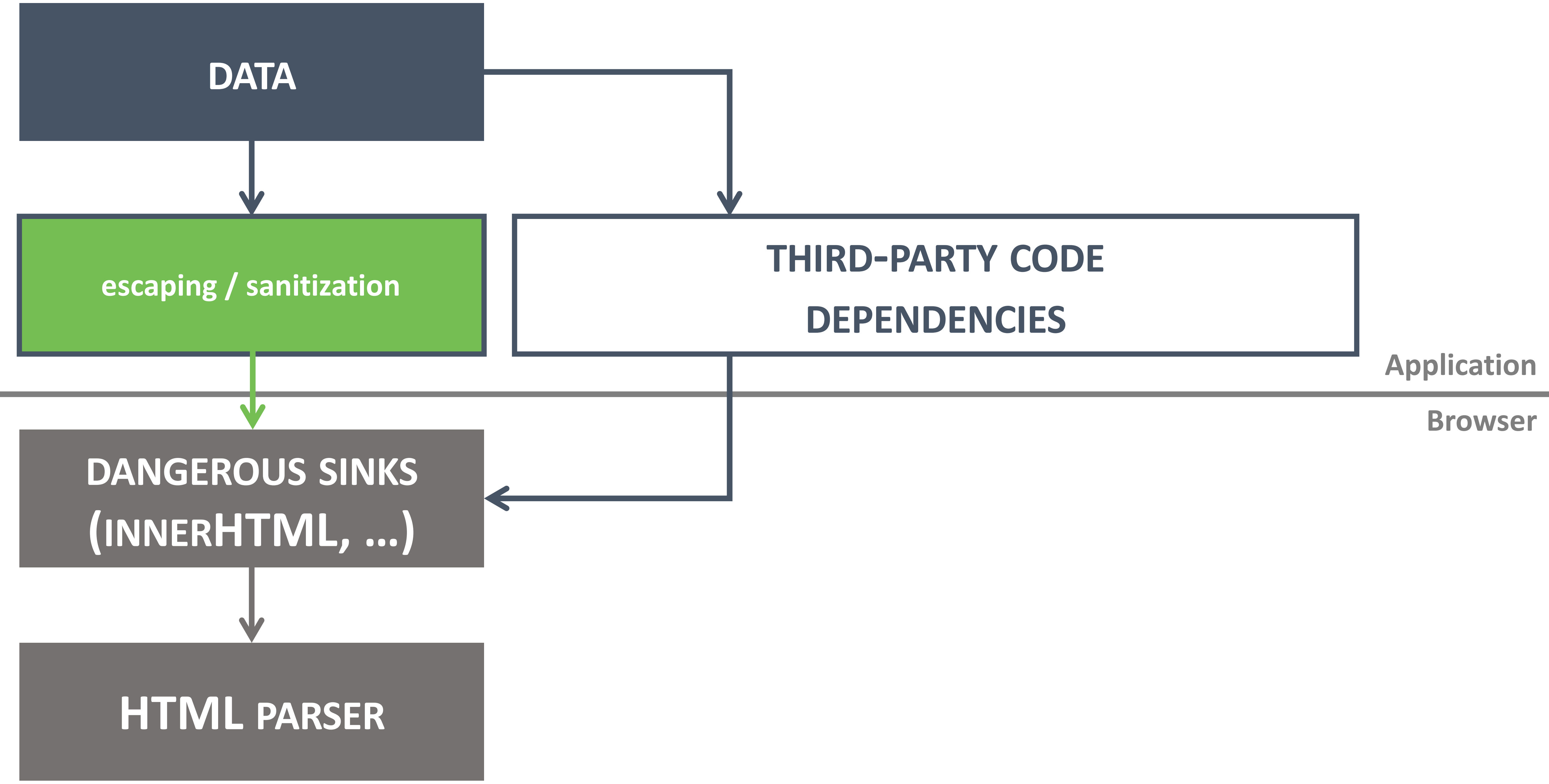
Enabling Trusted Types automatically results in better coding practices, even when only used in development

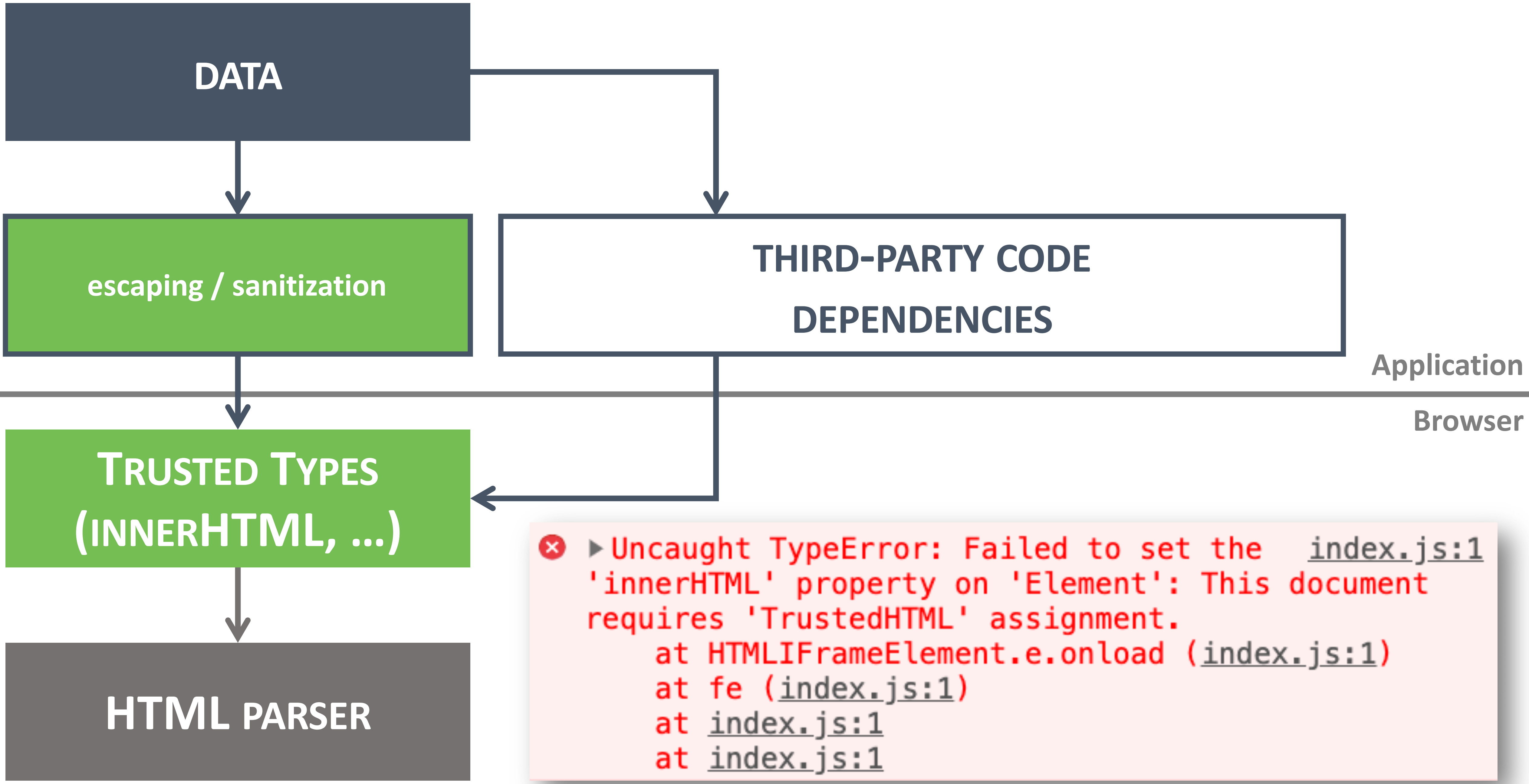
Trusted Types polyfills are available for non-supporting browsers

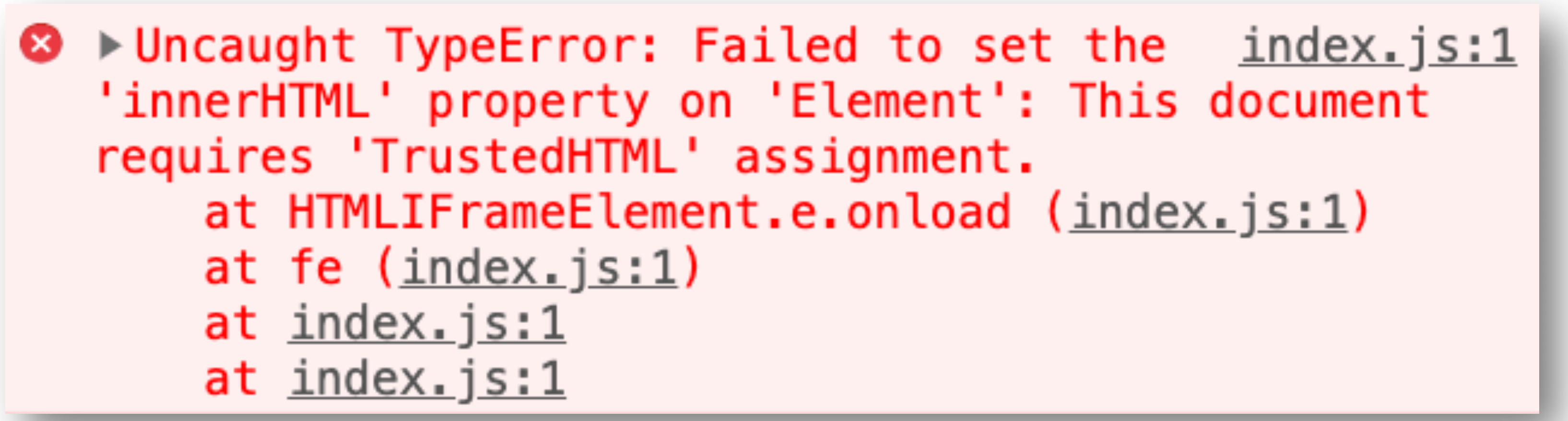
Trusted Types improves code security

Having Trusted Types point out unsafe assignments to the DOM helps fixing these issues in the application's code, benefiting all users


DATA
escaping / sanitization
THIRD-PARTY CODE DEPENDENCIES
Application
Browser
DANGEROUS SINKS (INNERHTML, …)
HTML PARSER


DATA
escaping / sanitization
THIRD-PARTY CODE
DEPENDENCIES
Application
Browser
DANGEROUS SINKS
(INNERHTML, …)
HTML PARSER





```
▸Uncaught TypeError: Failed to set the   index.js:1
'innerHTML' property on 'Element': This document
requires 'TrustedHTML' assignment.
    at HTMLIFrameElement.e.onload (index.js:1)
    at fe (index.js:1)
    at index.js:1
    at index.js:1
```

@PhilippeDeRyck

Enable trusted types by setting a CSP policy

```
Content-Security-Policy: require-trusted-types-for 'script'
```

Specify a default TT policy that is applied on all text assigned to HTML sinks

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/dompurify/2.2.4/purify.min.js"></script>
<script>
  //Define a default policy
  trustedTypes.createPolicy('default', {
    createHTML: (string, sink) =>
      DOMPurify.sanitize(string)
  });
</script>
```

Defining a default policy automatically applies the *createHTML* function on string-based assignments to *innerHTML*, which fixes the application

A default Trusted Types policy requires native browser support or the JS polyfill

Trusted Types is a browser-level measure

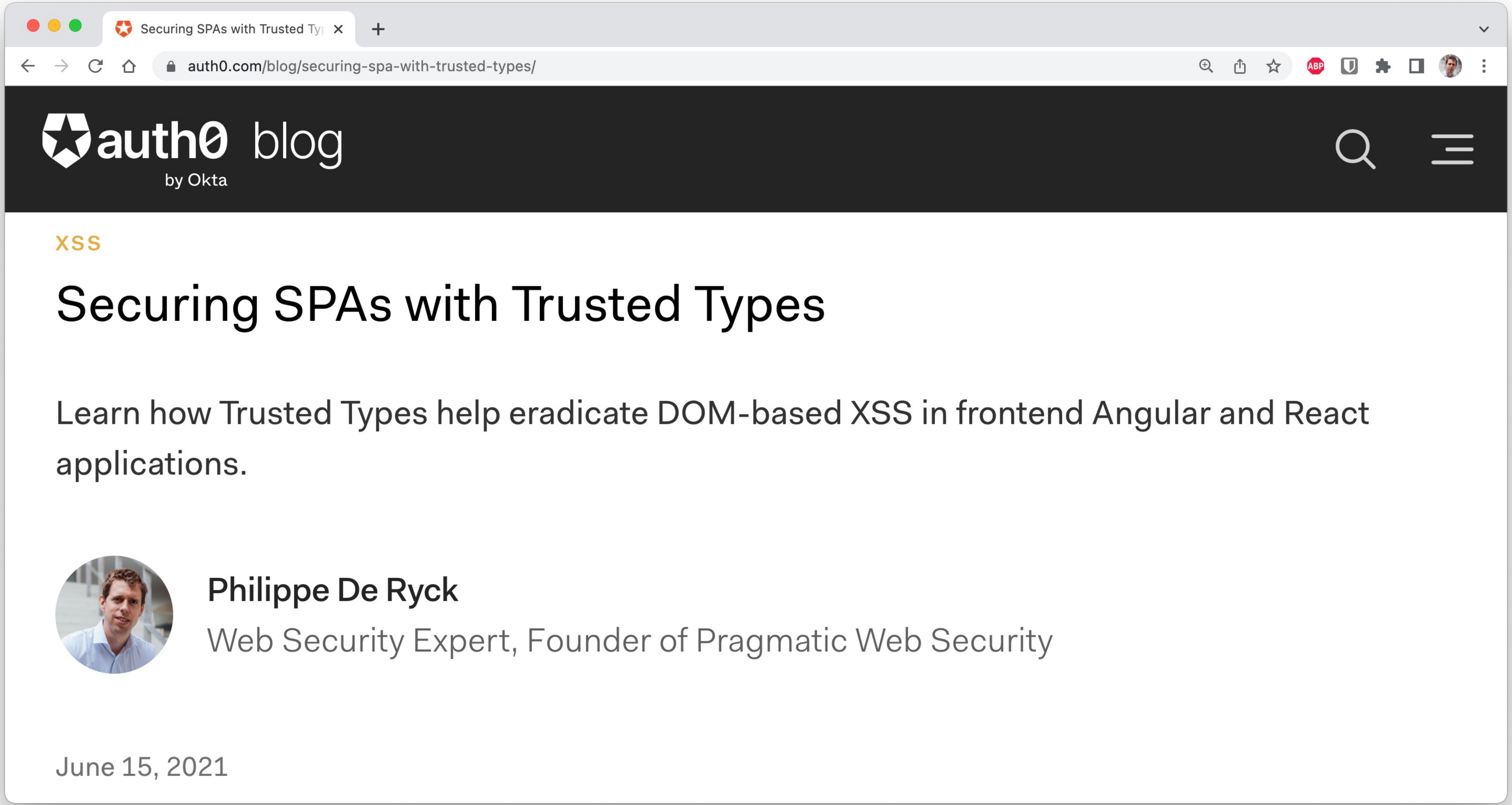
Trusted Types prevents that third-party code or dependencies from using dangerous sinks, and a default policy can automatically enable protection

Key takeaways

1. Angular supports Trusted Types out of the box

2. Enable TT in development to find insecure DOM assignments

3. Use a default policy when it is impossible to fix the actual code


Securing SPAs with Trusted Ty
auth0.com/blog/securing-spa-with-trusted-types/
auth0 blog
by Okta

XSS

Securing SPAs with Trusted Types

Learn how Trusted Types help eradicate DOM-based XSS in frontend Angular and React applications.

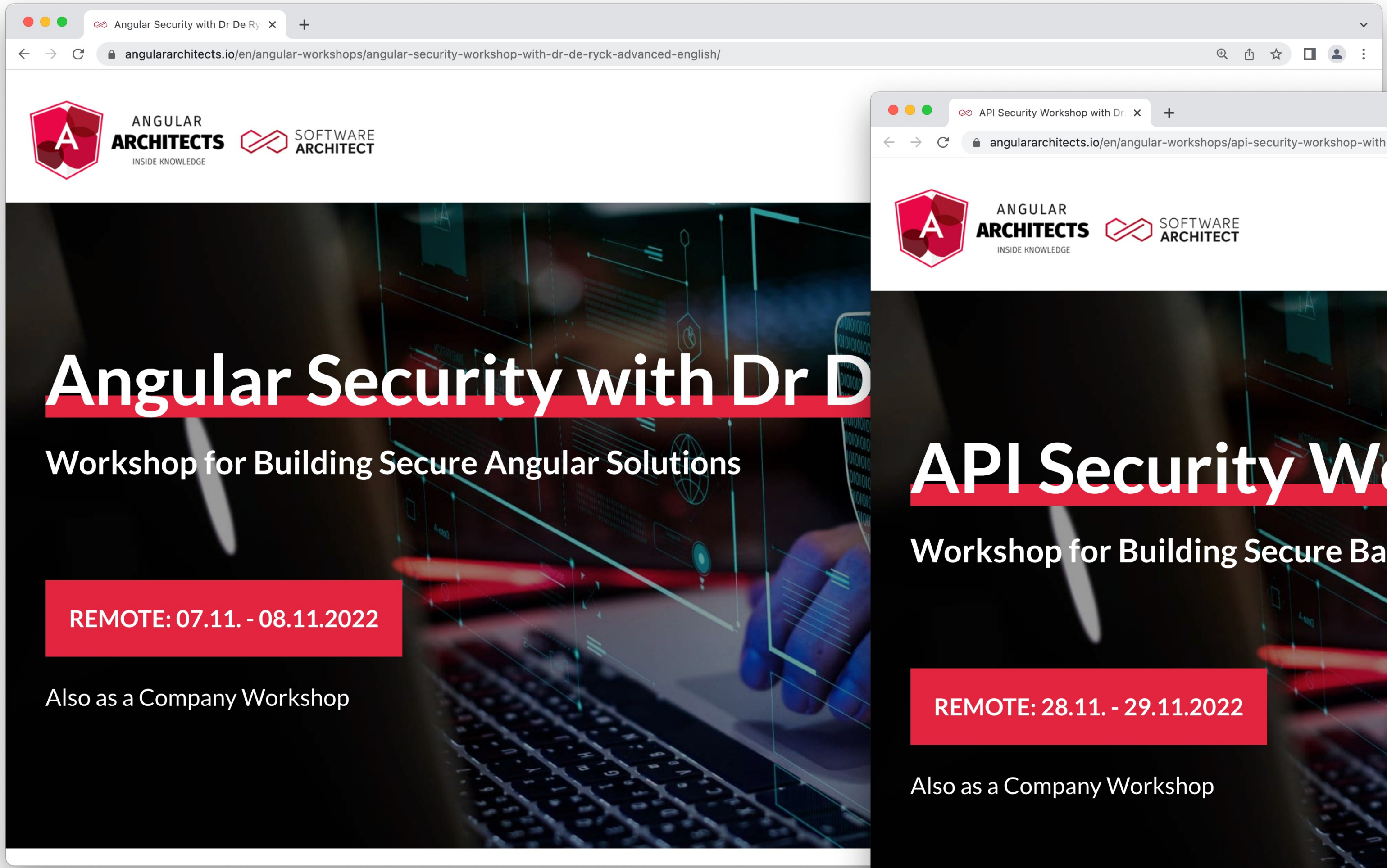
Philippe De Ryck

Web Security Expert, Founder of Pragmatic Web Security

June 15, 2021

https://auth0.com/blog/securing-spa-with-trusted-types/

Join our upcoming Angular / API security workshops!



Angular Security with Dr De Ry

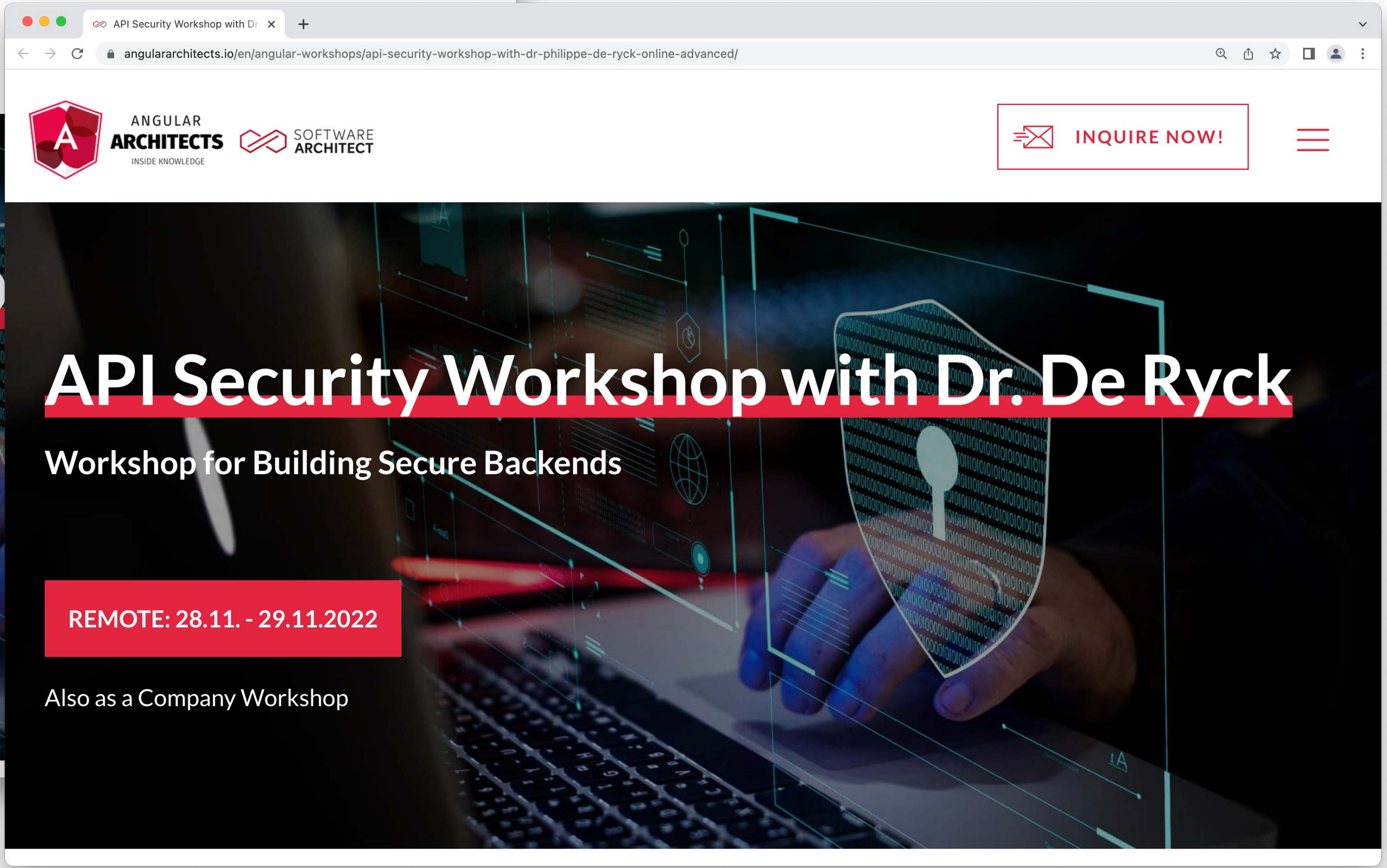
angulararchitects.io/en/angular-workshops/angular-security-workshop-with-dr-de-ryck-advanced-english/

Angular Security with Dr D

Workshop for Building Secure Angular Solutions

REMOTE: 07.11. - 08.11.2022

Also as a Company Workshop



API Security Workshop with Dr

angulararchitects.io/en/angular-workshops/api-security-workshop-with-dr-philippe-de-ryck-online-advanced/

INQUIRE NOW!

API Security Workshop with Dr. De Ryck

Workshop for Building Secure Backends

REMOTE: 28.11. - 29.11.2022

Also as a Company Workshop

HTTPS://PRAGMATICWEBSECURITY.COM/EVENTS

Thank you!

Connect on social media
to stay in touch on security

@PhilippeDeRyck

/in/PhilippeDeRyck