Exploring Blazor Feature Enhancements

Alex Wolf
.NET Developer

www.thecodewolf.com

Blazor Improvements in ASP.NET 6.0

Simpler features

- Component parameters
- Error handling
- HTML document manipulation

- Dynamic components
- JavaScript interoperability
- Ahead of time compilation (AoT)

More involved

Demo

Binding component parameters via the URL

Demo

Improving error handling in the UI

Demo

Modifying the HTML document

Understanding Dynamic Components

The Dynamic Component

YourComponent.cs

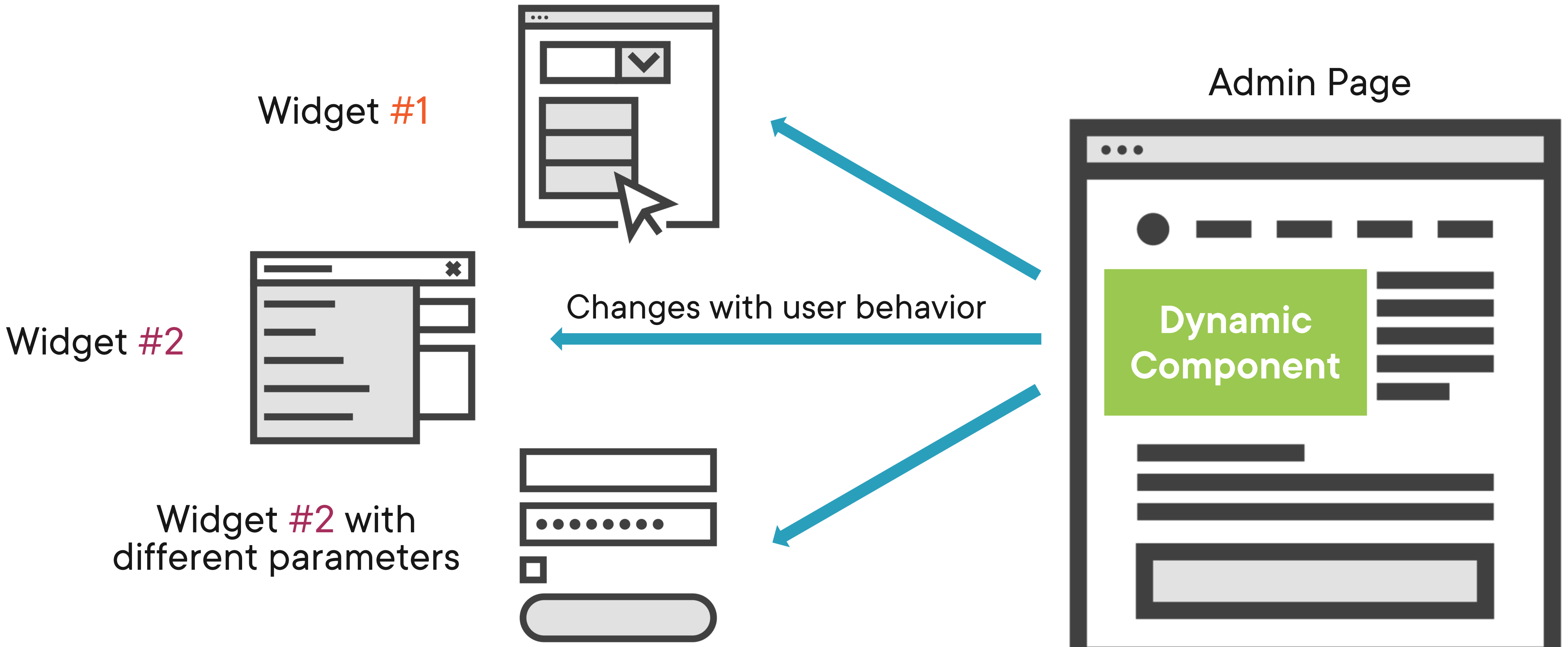
```
<DynamicComponent Type="@componentType" Parameters="@parameters" />
```

The Dynamic Component can programmatically render other components

The Type parameter specifies the C# Type of the component to render

The Parameters parameter allows us to pass values down into the rendered component

Dynamic Component Examples


Widget #1
Widget #2
Widget #2 with different parameters
Changes with user behavior
Admin Page
Dynamic Component

Dynamic Component Use Cases

User selection

The user decides which component they need

Unstructured data

Displaying data from loosely structured sources

Search results

Display different types of results depending on criteria

Demo

Working with Dynamic Components

Exploring JavaScript Interoperability

Essential New JavaScript Features

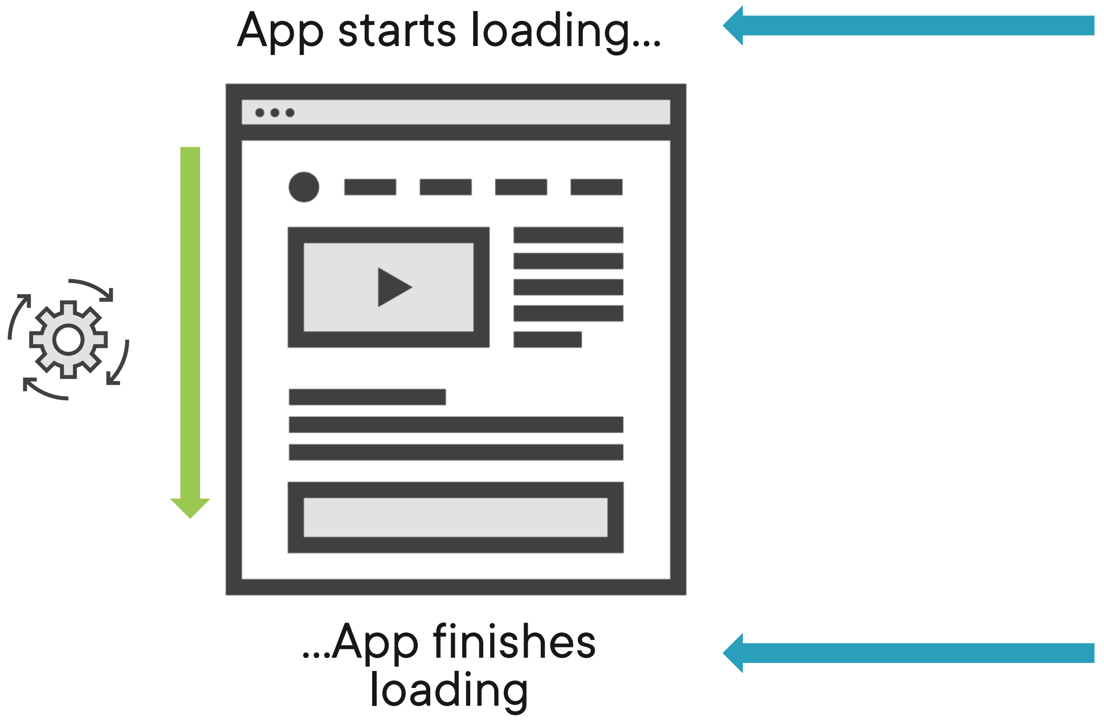
JavaScript Initializers

Execute code before and after the Blazor app loads

Rendering Blazor components via JS

Add Blazor components to hybrid or JavaScript apps

Utilizing JavaScript Initializers



```
function beforeStart() {
        // your code here
}

function afterStarted() {
        // your code here
}
```

JavaScript module

Rendering Blazor Components with JavaScript

```
builder.RootComponents.RegisterForJavaScript<HelloWor
ld>(identifier: "helloworld");
```

Register the Hello World Blazor component for JavaScript in program.cs

```
let element =
document.getElementById(‘helloworld’););

await blazor.rootComponents.add(element,
‘helloworld’);, {});
```

Use JavaScript to retrieve an HTML element and inject the Blazor component

Additional JavaScript Interop Features

.NET to JavaScript streaming

Improved byte array and data object performance

Custom Blazor HTML elements and framework wrappers

Demo

Rendering Blazor components with JavaScript Initializers

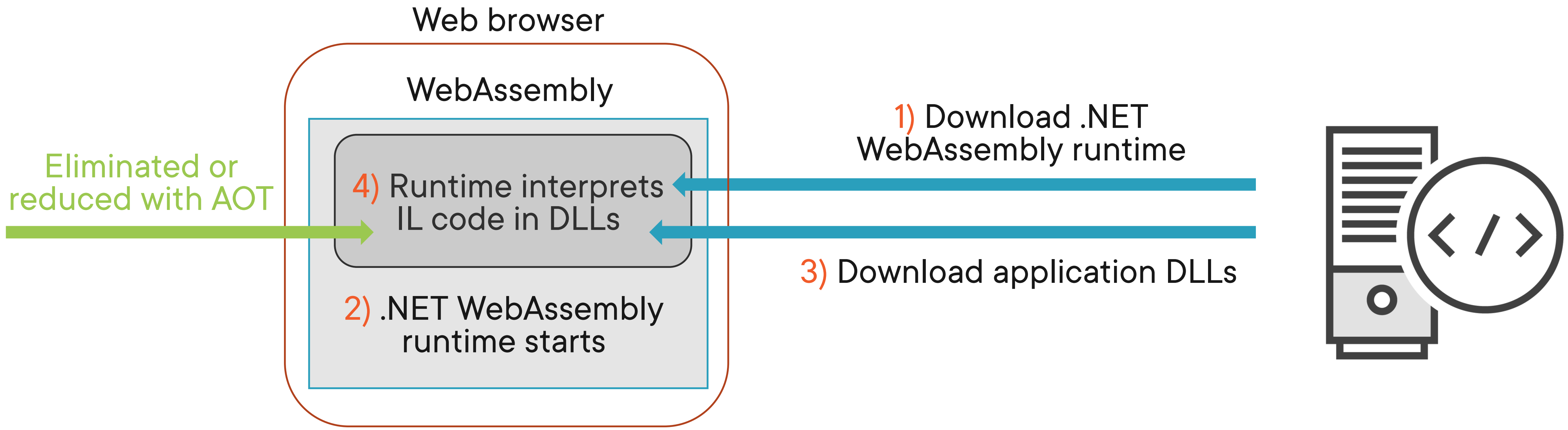
Understanding Ahead-of-Time Compilation

Ahead-of-Time Compilation (AOT)

Precompiles Blazor apps instead WebAssembly for improved performance in the browser

(Blazor WebAssembly hosting model only)

Blazor WebAssembly Execution


Web browser
WebAssembly
Eliminated or reduced with AOT
4) Runtime interprets IL code in DLLs
2) .NET WebAssembly runtime starts
1) Download .NET WebAssembly runtime
3) Download application DLLs

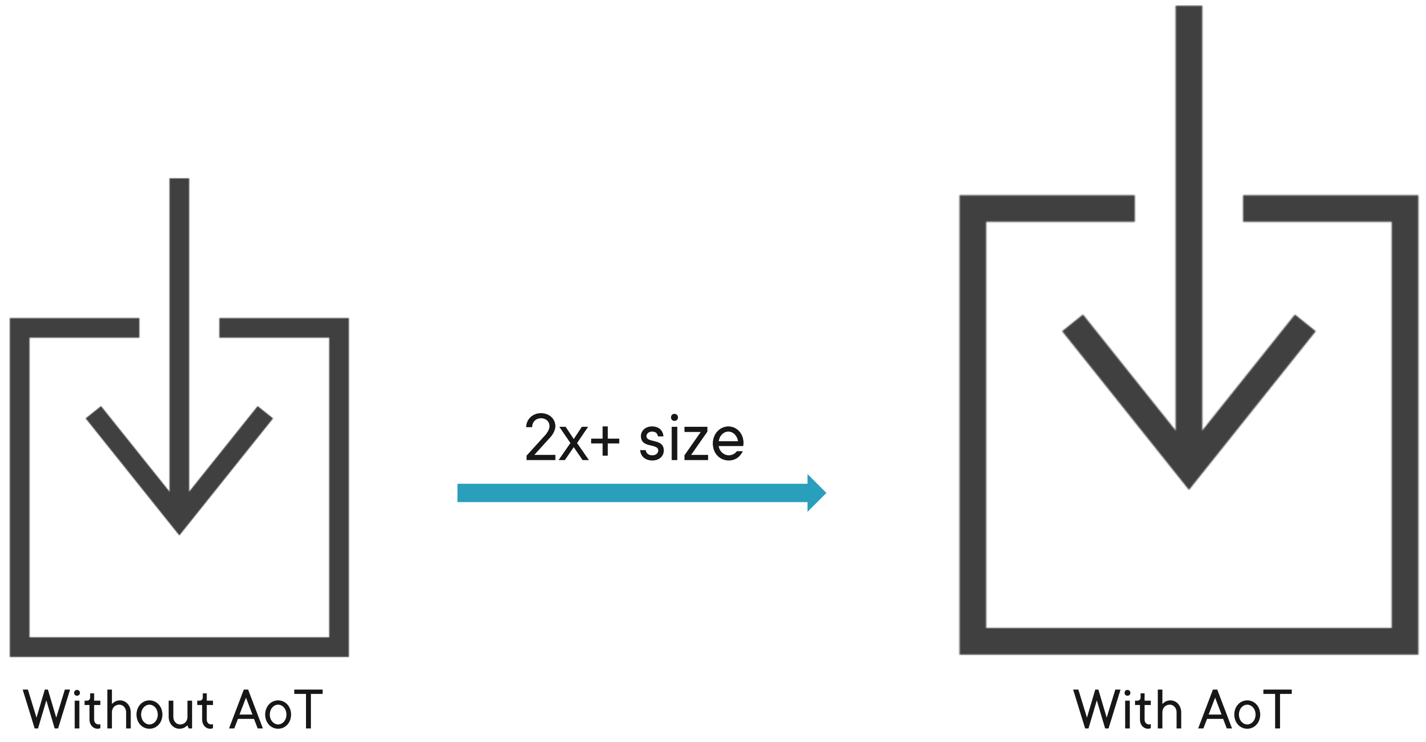
Ideal Cases for AOT Compilation

Image editing

Games and renderings

Complex algorithms

Blazor WebAssembly Download Size



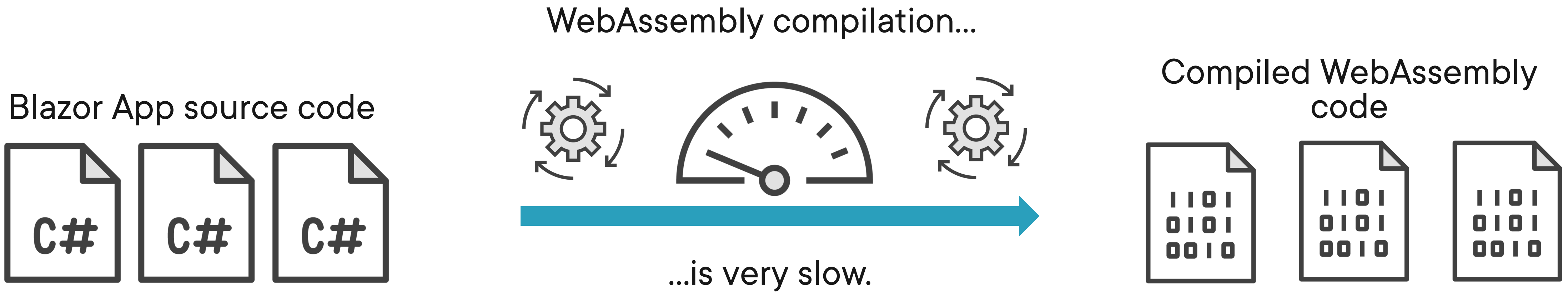
Performance Decisions

Consider the way your app is used

AOT is not always the answer

Other features might provide better solutions

AOT Compilation Considerations



Other Blazor Performance Considerations

Many internal framework optimizations

Improvements to runtime relinking that remove unnecessary code

Decreased download sizes (outside of AOT)

Demo

Applying Ahead of Time Compilation

Overview/ Summary

- Blazor introduces many quality of life improvements in .NET 6.0
- Component parameters can now be populated from the URL and marked as required
- Error Boundaries provide better exception handling for the user
- Blazor can now easily manipulate the HTML document head and title
- The Dynamic Component allows us to programmatically render components by type
- Blazor components can now be rendered via JavaScript
- JavaScript initializers let us run code before and after the Blazor app loads
- AoT Compilation greatly improves performance in some scenarios, with certain drawbacks