The Angular 2 Compiler

Lessons learned

Tobias Bosch
@tbosch1009

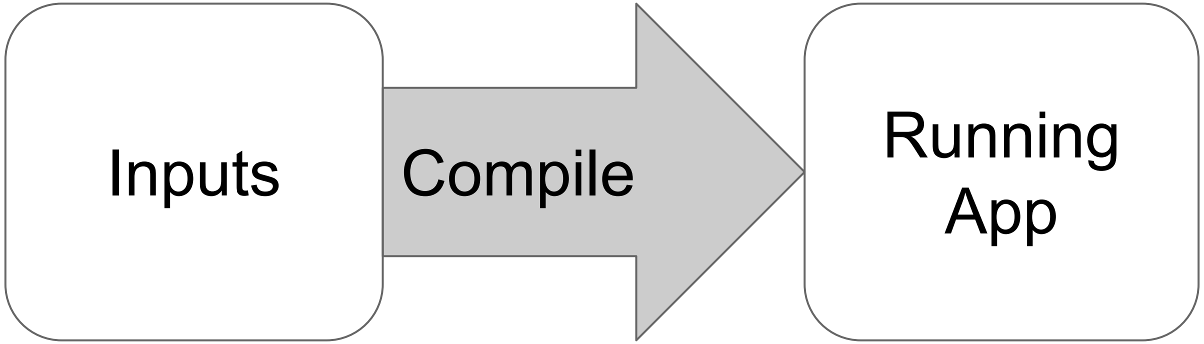
ANGULAR CONNECT

This talk is about...

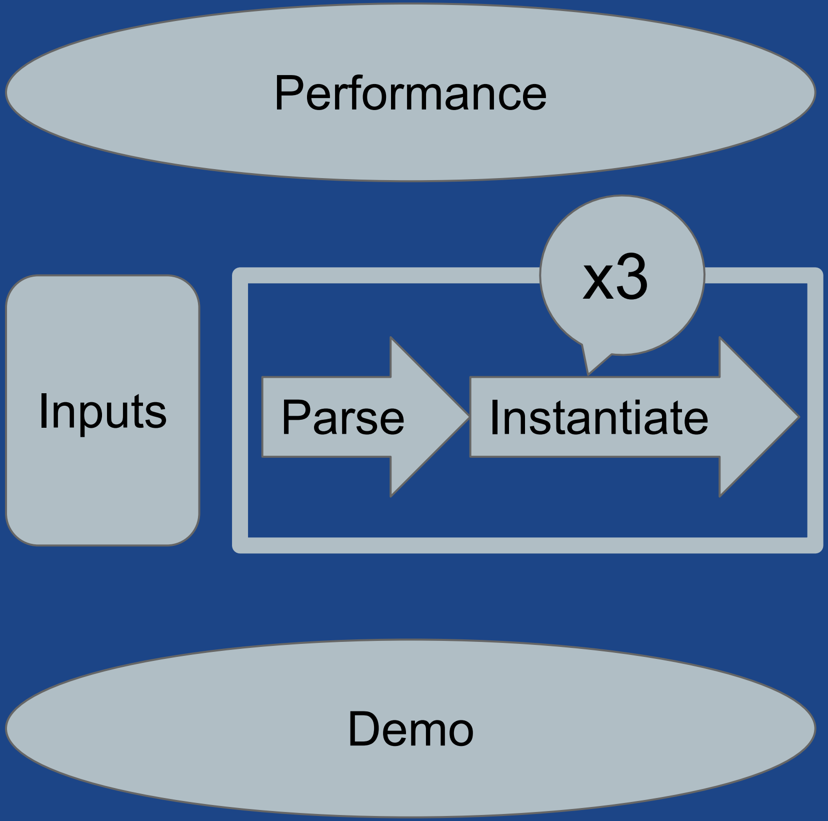
... How Angular 2 works internally.

... Our journey to get there.

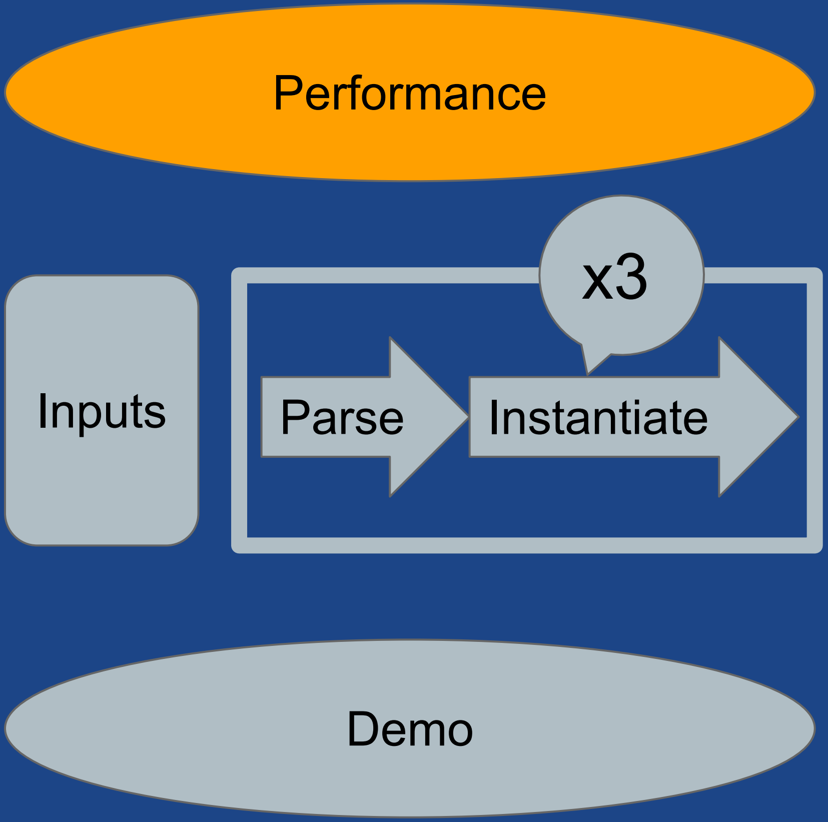
The Angular 2 Compiler



Agenda



Performance



Performance - What is fast?

Page Load

Switching a Route

Update Values

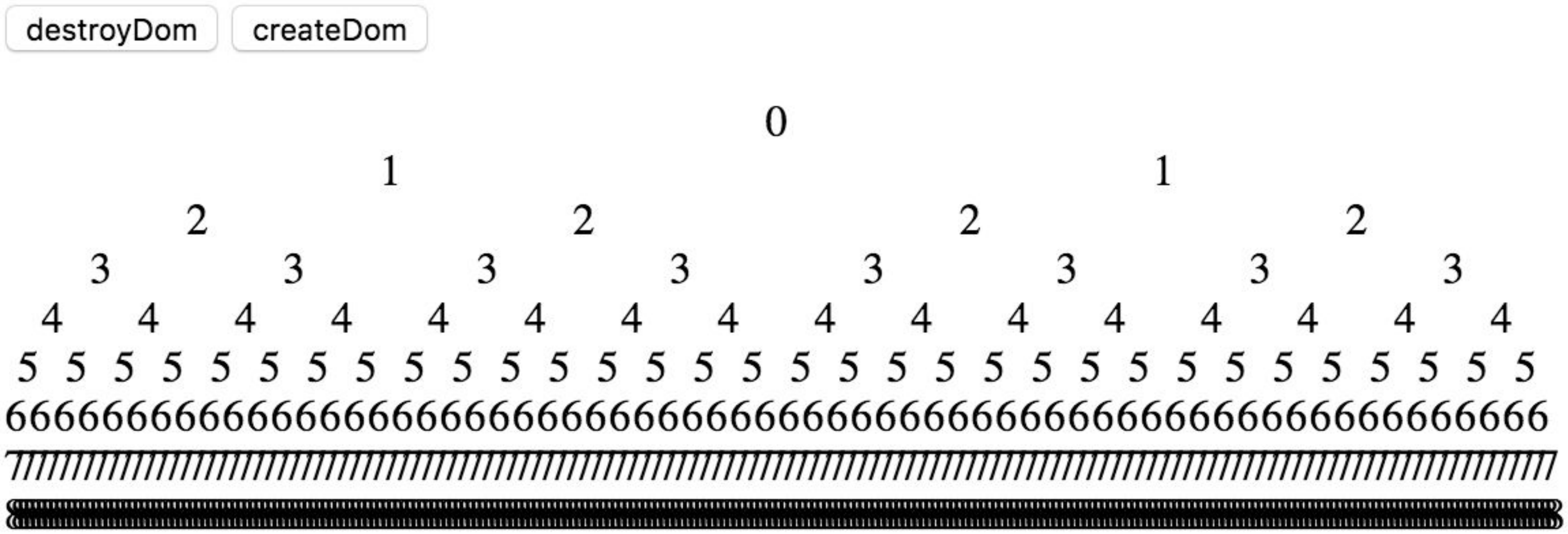
Performance - What is fast?

Page Load

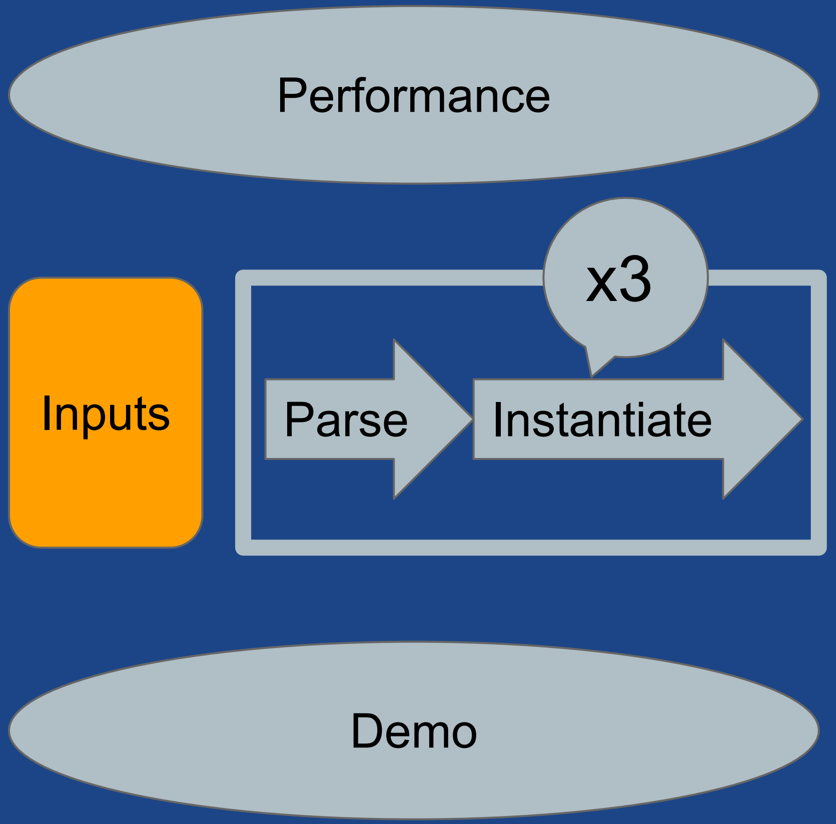
Switching a Route

Update Values

Performance - Deep Tree Benchmark Application



Inputs



Inputs - Component Metadata

```
@Component({
  templateUrl: 'hello_comp.html'
})
class HelloComp {
  user = { name: 'Tobias' };
}
```

Inputs - Component Templates

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Component

```
@Component({ ... })
class HelloComp {
  user = { name: 'Tobias' };
}
```

Inputs - Component Templates

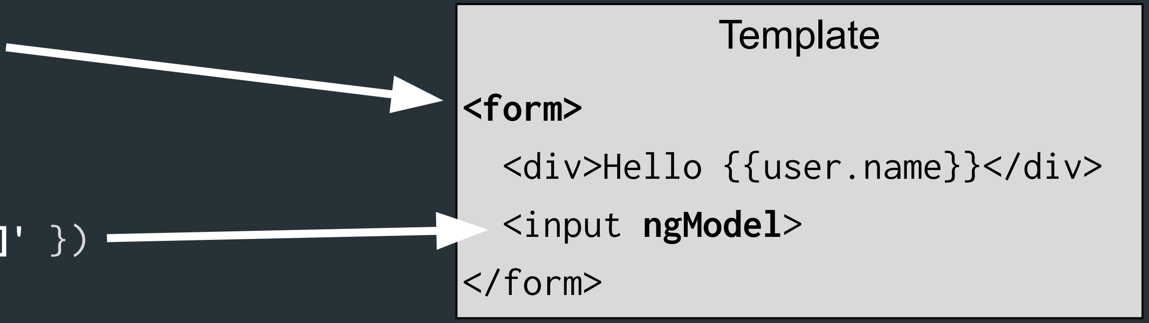
```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Component

```
@Component({ ... })
class HelloComp {
  user = { name: 'Tobias' };
}
```

Inputs - Directive Metadata

```
@Directive({ selector: 'form' })
class NgForm {...}

@Directive({ selector: '[ngModel]' })
class NgModel { ... }
```

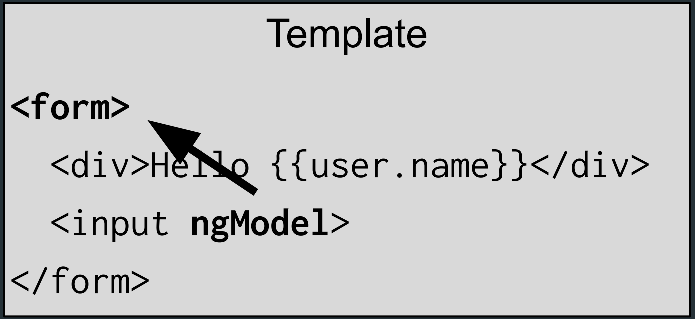


Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Inputs - Directive Dependencies

```
@Directive({ selector: 'form' })
class NgForm {...}

@Directive({ selector: '[ngModel]' })
class NgModel {
  ...
  constructor(form: NgForm) {...}
}
```


Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Inputs - There are more...

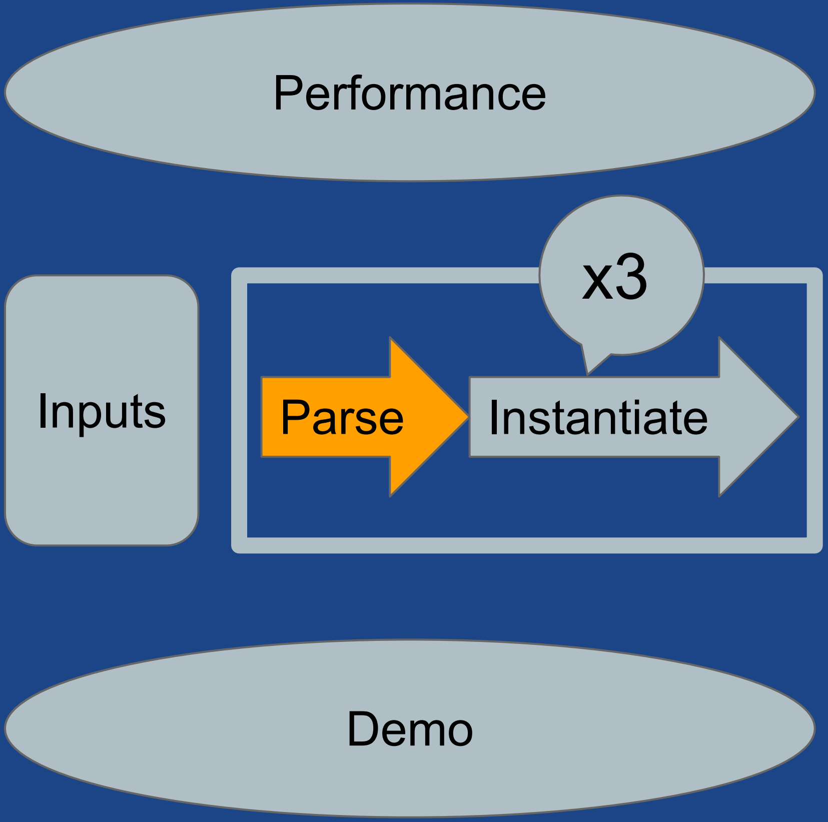
@NgModule, @Pipe

@Input, @Output, @ContentChildren, @ViewChildren, @HostBinding, @HostListener

<template>

...

Template Parser



Template Parser - HTML

```
{ name: 'form', children: [
  { name: 'div', children: [
    { text: 'Hello ' },
    { text: '' },
  ]},
  { name: 'input',
    attrs: [['ngModel', '']]
  }
]}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Template Parser - Expressions

```
...
{ text: '',
  expr: {
    propPath: ['user', 'name'] },
    line: 2, col: 14
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Template Parser - Line/Column Numbers

```
Uncaught EXCEPTION: Error in hello_comp.html:2:14
  Uncaught TypeError: Cannot read property 'name'
    of undefined...
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

AST

```
{ text: '',
  expr: {
    propPath: ['user', 'name'] },
    line: 2, col: 14
  }
}
```

Template Parser - Directives

```
{ name: 'input', attrs: [['ngModel', '']],
  directives: [{
    ctor: NgModel,
    deps: [NgForm],
  }]
}
```

Template

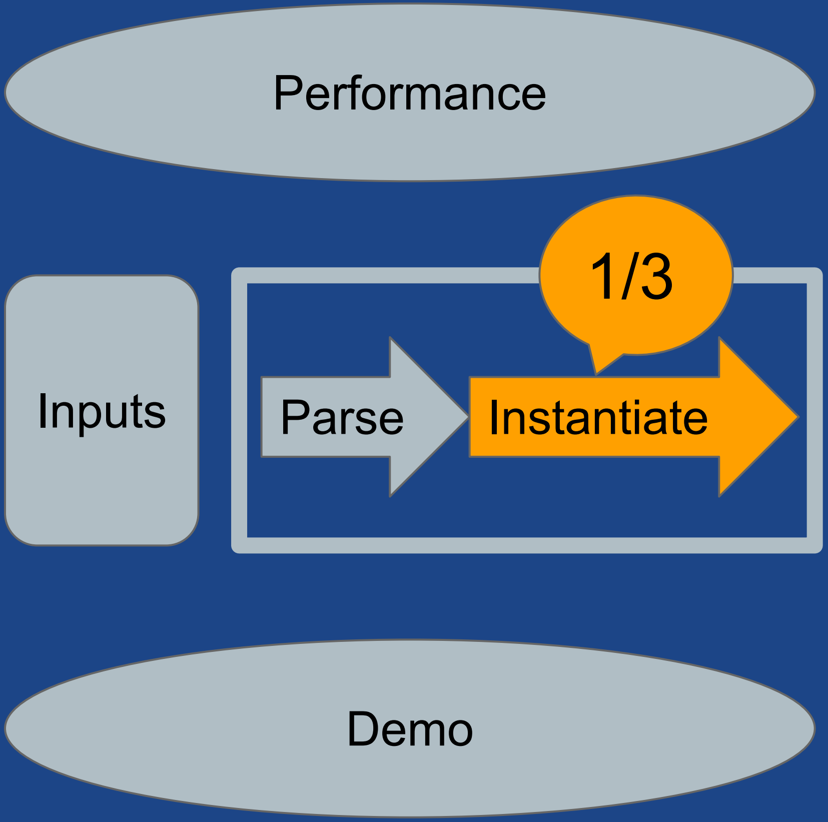
```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel required>
</form>
```

Directive

```
@Directive({
    selector: '[ngModel]' })
class NgModel {
  ...
  constructor(form: NgForm) {...}
}
```

Template Instantiation 101

A Simple Approach



Instantiation 101

Fastest way to create a DOM element?

- ✅ element.innerHTML = '<form>...'
- ✅ var clone = element.cloneNode(true)
- ✅ document.createElement(...)

Instantiation 101 - element.innerHTML

```
var compEl = ...;
compEl.innerHTML = '<form>...';

var userNameText = compEl.childNodes[0]
    .childNodes[0].childNodes[0];
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Instantiation 101 - document.createXYZ

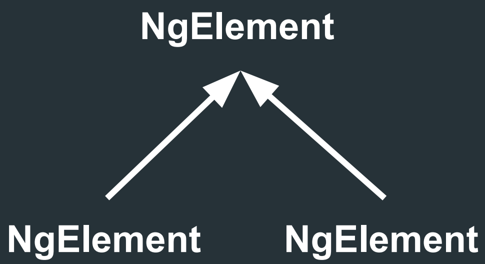
```
var divEl = document.createElement('div');
divEl.appendChild(document.createTextNode('Hello'));

var userNameText = document.createTextNode();
divEl.appendChild(userNameText);
...
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Instantiation 101 - Create Elements

```
class NgElement {
  parent: NgElement;

  domEl: HTMLElement;
  directives: Map; // type -> instance
}
```



Instantiation 101 - Create Elements

```
class NgElement {
  parent: NgElement;
  domEl: HTMLElement;
  directives: Map; // type -> instance

  constructor(parent: NgElement, ast: ElementAst) {
    this.domEl = document.createElement(ast.name);
    ast.attrs.forEach( (attr) =>
        this.domEl.setAttribute(attr[0], attr[1]));
    parent.domEl.appendChild(this.domEl);
    // ... create the directives
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Element AST

```
{ name: 'input', directives: ...,
  attrs: [['ngModel', '']]
}
```

Instantiation 101 - Create Directives

```
class NgElement {
  parent: NgElement;
  domEl: HTMLElement;
  directives: Map; // type -> instance

  createDirective(dirAst) {
    var deps = ast.deps.map( (depType) =>
      this.getDirectiveDep(depType);

    this.directives.set(ast.ctor,
      new ast.ctor(...deps));
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Element AST

```
{ name: 'input', directives: [
  { ctor: NgModel,
    deps: [NgForm] }
], attrs: ...}
```

Instantiation 101 - Resolve Dependencies

```
class NgElement {
  parent: NgElement;
  domEl: HTMLElement;
  directives: Map; // type -> instance

  getDirectiveDep(dirType) {
    if (this.directives.has(dirType)) {
      return this.directives.get(dirType);
    }
    return this.parent.getDirectiveDep(dirType);
  }
}
```

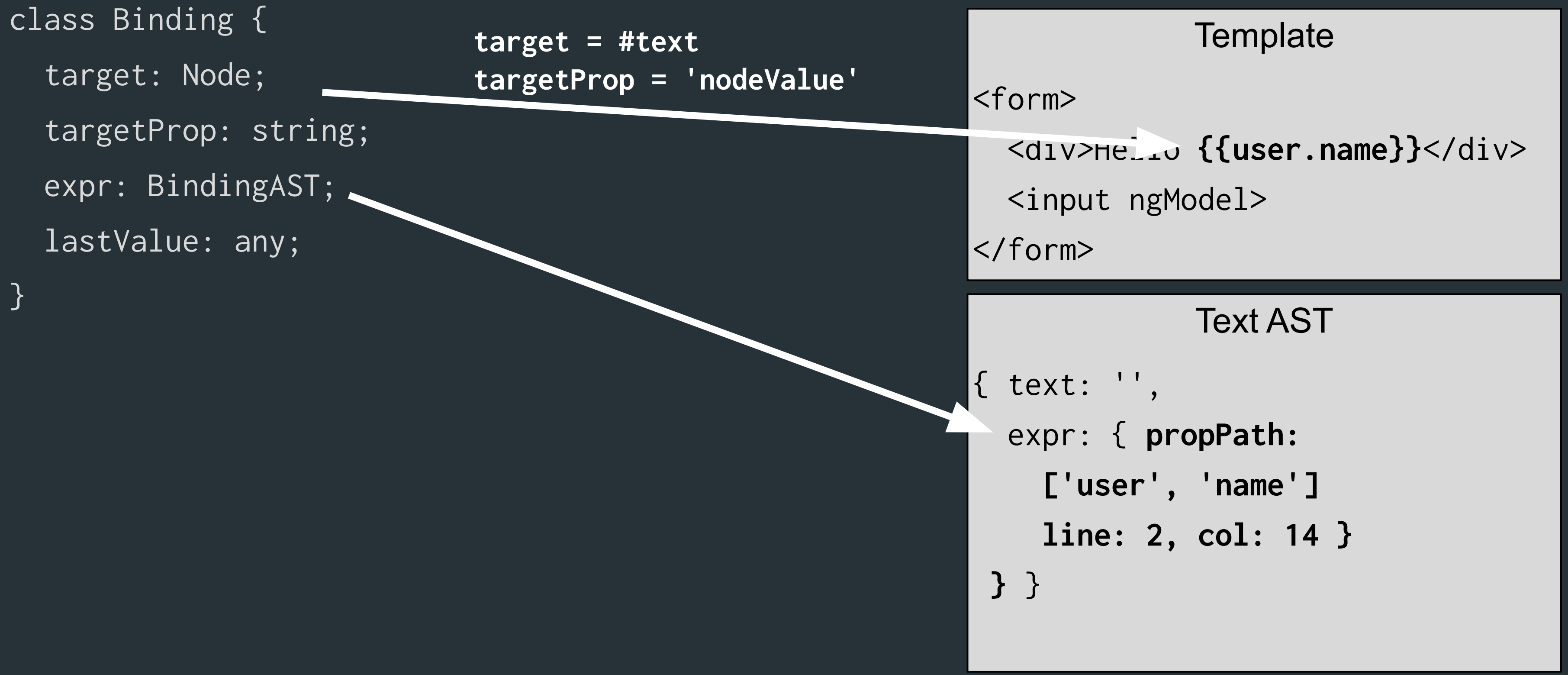
Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Element AST

```
{ name: 'input', directives: [
  { ctor: NgModel,
    deps: [NgForm] }
], attrs: ...}
```

Instantiation 101 - Data Binding



```
class Binding {
  target: Node;
  targetProp: string;
  expr: BindingAST;
  lastValue: any;
}
```

```
target = #text
targetProp = 'nodeValue'
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Text AST

```
{ text: '',
  expr: { propPath:
    ['user', 'name']
    line: 2, col: 14 }
 } }
```

Instantiation 101 - Data Binding

```
class Binding {
  target: Node;
  targetProp: string;
  expr: BindingAST;
  lastValue: any;
  check(component: any) {
    var newValue = evaluate(this.expr, component);
    if (newValue !== this.lastValue) {
      this.target[this.targetProp] =
        this.lastValue = newValue;
    }
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Text AST

```
{ text: '',
  expr: { propPath:
    ['user', 'name']
    line: 2, col: 14 }
 } }
```

Instantiation 101 - Data Binding Error Handling

```
class Binding {
  check(component: any) {
    try {
      ...
    } catch (e) {
      throw new Error(`Error in ${this.expr.line}:`+
        `${this.expr.col}: ${e.message}`);
    }
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Text AST

```
{ text: '',
  expr: { propPath:
    ['user', 'name']
    line: 2, col: 14 }
 } }
```

Instantiation 101 - View

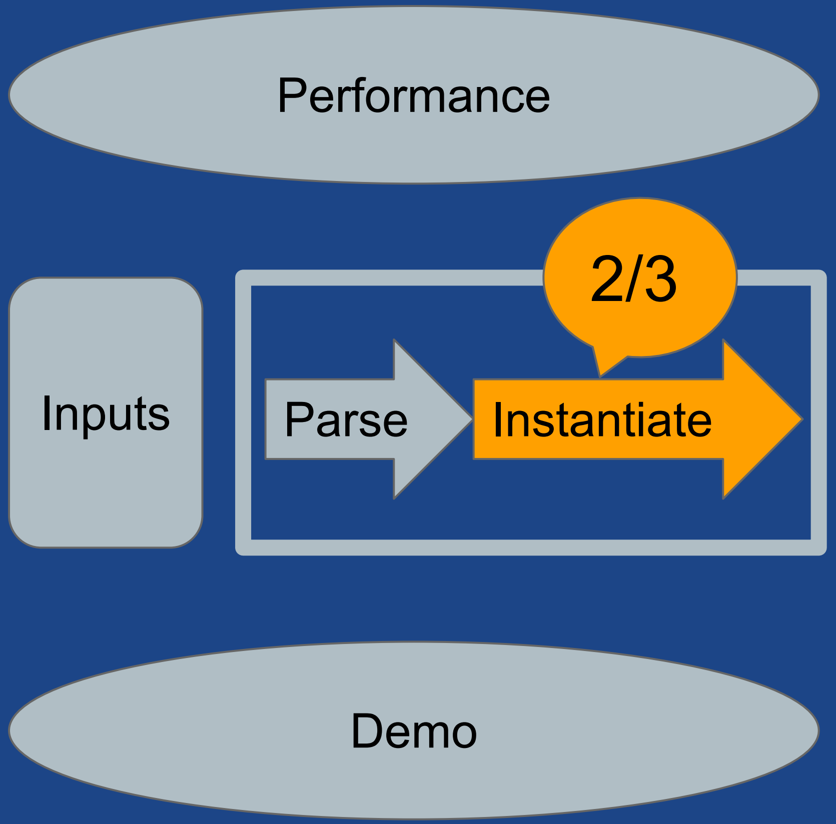
```
class View {
  component: any;
  ngElements: NgElement[];
  bindings: Binding[];

  dirtyCheck() {
    this.bindings.forEach( (binding) => binding.check(this.component) );
  }
}
```

Instantiation 101 - Performance

Deep Tree benchmark:
About the same as Ng 1

Template Instantiation 201

Optimize data structures



Instantiation 201 - Previous NgElement

```
class NgElement {
  parent: NgElement;
  domEl: HTMLElement;
  directives: Map; // type -> instance

  getDirectiveDep(dirType) {
    if (this.directives.has(dirType)) {
      return this.directives.get(dirType);
    }
    return this.parent.getDirectiveDep(dirType);
  }
}
```

Instantiation 2 - Optimized Data Structures

```
class InlineNgElement {
  ...
  dir0, dir1, ...: any;
  dirType0, dirType1, ...: any;

  getDirectiveDep(dirType) {
    if (type === this.dirType0) return this.dir0;
    if (type === this.dirType1) return this.dir1;
    ...
  }
}
```

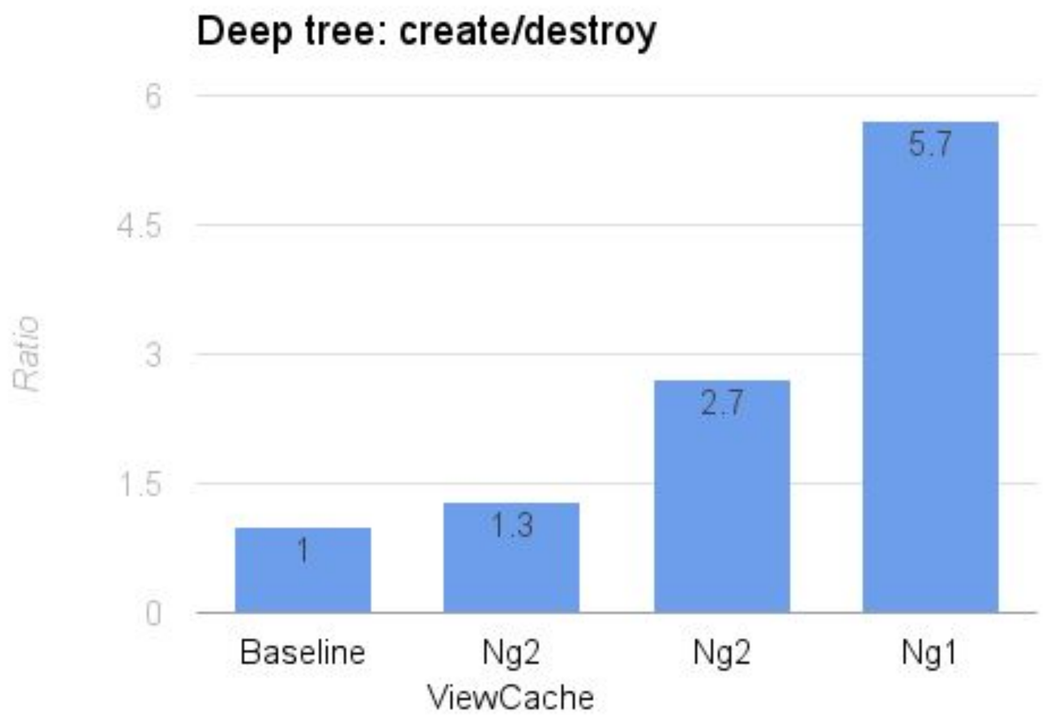
{ **Fast Property Access via Hidden Classes!** }

Instantiation 201 - View Cache

```
var cache: View[];
class View {
  ngElements: NgElement[];
  bindings: Binding[];
  dehydrate() {
      // keep the DOM node!
      // clear directive instances
      // clear last binding values
  }
  hydrate() {
    // create directive instances again
  }
}
```

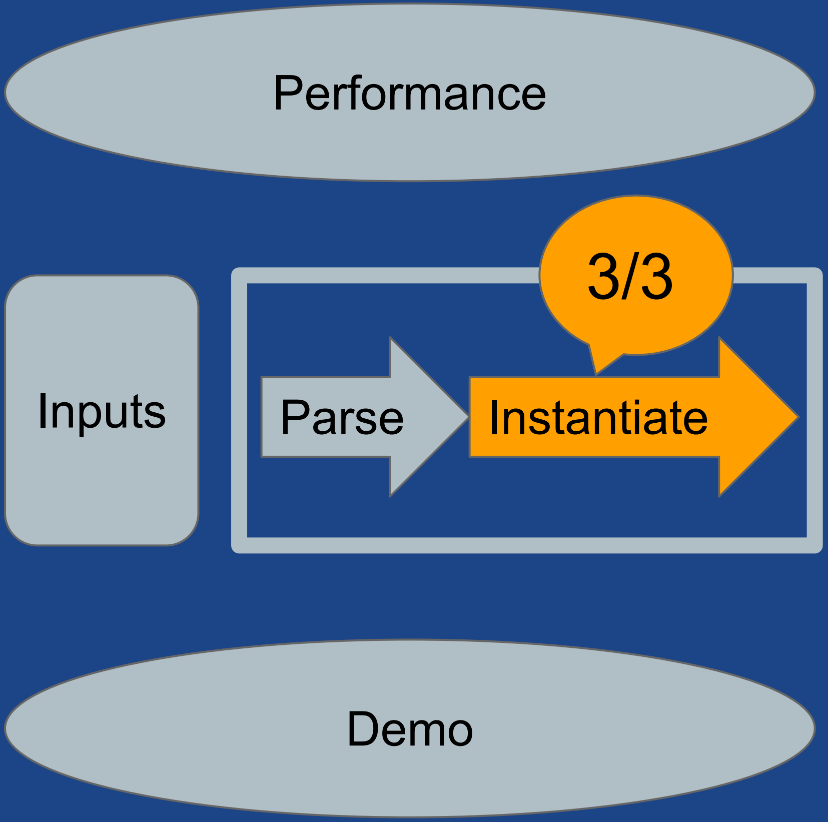
{ **Reuse Data Structures!** }

Instantiation 201 - Performance



Template Instantiation 301

Generate view classes



Instantiation 301 - Previous View Class

```
class View {
  component: any;
  ngElements: InlineNgElement[];
  bindings: Binding[];
}

class InlineNgElement {
  dir0, dir1, ...: any;
  dirType0, dirType1, ...: any;
 ...
}
```

Instantiation 301 - Create Elements

```
function HelloCompView(component) {
  this.component = component;
  this.node0 = document.createElement('form');
  this.node1 = document.createElement('div');
  this.node0.appendChild(this.node1);
  ...
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Instantiation 301 - Create Directives

```
function HelloCompView(component) {
  ...
  this.dir0 = new NgForm();
  ...
  this.dir1 = new NgModel(this.dir0);
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input ngModel>
</form>
```

Instantiation 301 - Data Binding

```
HelloCompView.prototype.dirtyCheck = function() {
  var v = this.component.user.name;
  if (v !== this.exprVal0) {
    this.node3.ngModel = v;
    this.exprVal0 = v;
  }
}
```

Template

```
<form>
  <div>Hello {{user.name}}</div>
  <input [(ngModel)]="user.name">
</form>
```

Instantiation 301 - Generated View Classes

```
class HelloCompView {
  node0, node1, node2, ... : Node;
  dir0, dir1, dir2, ...: any;
  exprVal0, exprVal1, exprVal2, ...: any;
}
```

{ Fast Property Access everywhere! }

Instantiation 301 - How to generate the View Classes?

Instantiation 101	Instantiation 301
Directives / DOM nodes: Create / call them	Generate code that will create / call them
Directive / DOM node references	Property names on View class

Instantiation 301 - How to generate the View Classes?

```
class NgElement {
  domEl: HTMLElement;

  constructor(parent: NgElement,
              ast: ElementAst) {

    this.domEl =
      document.createElement(ast.name);
    parent.domEl.appendChild(
      this.domEl);
    // ...
  }
}
```

```
class CompileElement {
  domElProp: string = newPropertyVar();
  stmts: string[];
  constructor(parent: CompileElement,
              ast: ElementAst) {
    this.stmts = [`
      this.${domElProp} =
        document.createElement('${ast.name}');
      this.${parent.domElProp}.appendChild(
        ${this.domElProp});
    `]; // ...
  }
}
```

Instantiation 301 - How to generate the View Classes?

```
class NgElement {
  domEl: HTMLElement;

  constructor(parent: NgElement,
              ast: ElementAst) {

    this.domEl =
      document.createElement(ast.name);
    parent.domEl.appendChild(
      this.domEl);
    // ...
  }
}
```

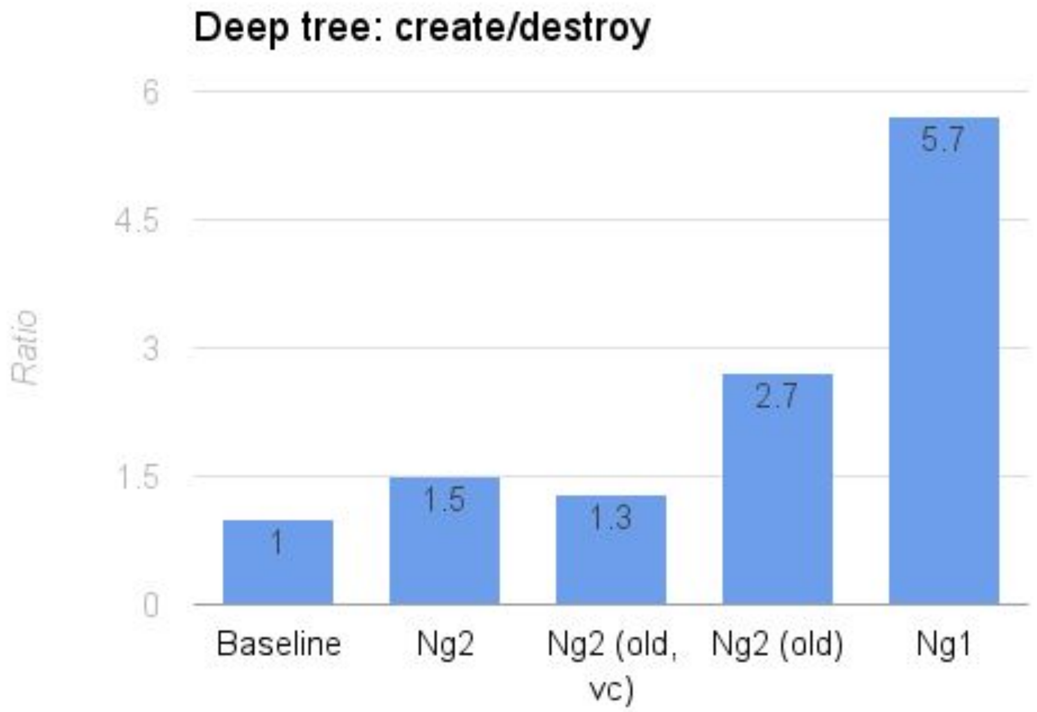
```
class CompileElement {
  domElProp: string = newPropertyVar();
  stmts: string[];
  constructor(parent: CompileElement,
              ast: ElementAst) {
    this.stmts = [`
      this.${domElProp} =
        document.createElement('${ast.name}');
      this.${parent.domElProp}.appendChild(
        ${this.domElProp});
    `]; // ...
  }
}
```

Instantiation 301 - How to generate the View Classes?

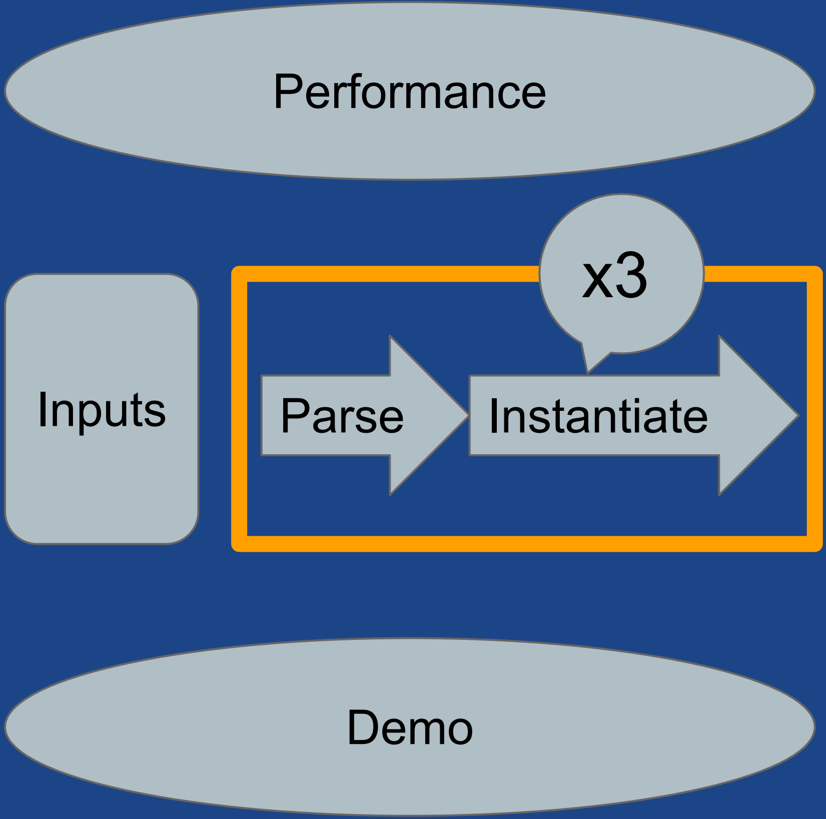
```
class NgElement {
  domEl: HTMLElement;

  constructor(parent: NgElement,
              ast: ElementAst) {

    this.domEl =
       document.createElement(ast.name);
    parent.domEl.appendChild(
      this.domEl);
    // ...
  }
}
```

```
class CompileElement {
  domElProp: string = newPropertyVar();
  stmts: string[];
  constructor(parent: CompileElement,
              ast: ElementAst) {
    this.stmts = [`
      this.${domElProp} =
        document.createElement('${ast.name}');
      this.${parent.domElProp}.appendChild(
        ${this.domElProp});
    `]; // ...
  }
}
```

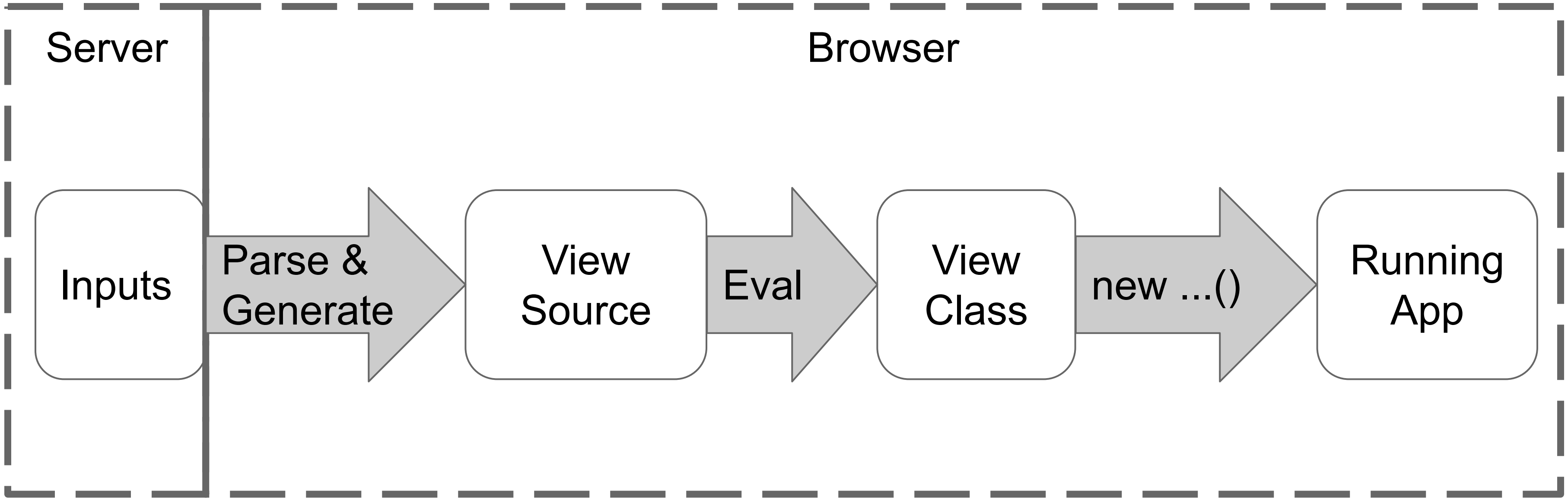
Instantiation 301 - Performance



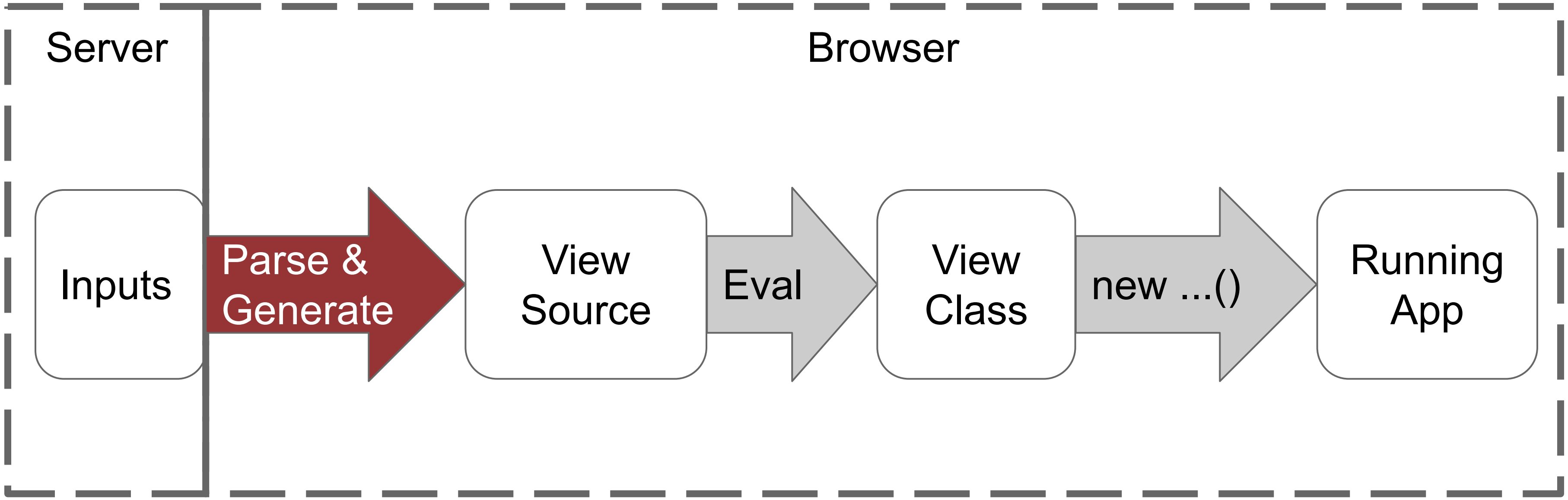
JIT Compilation



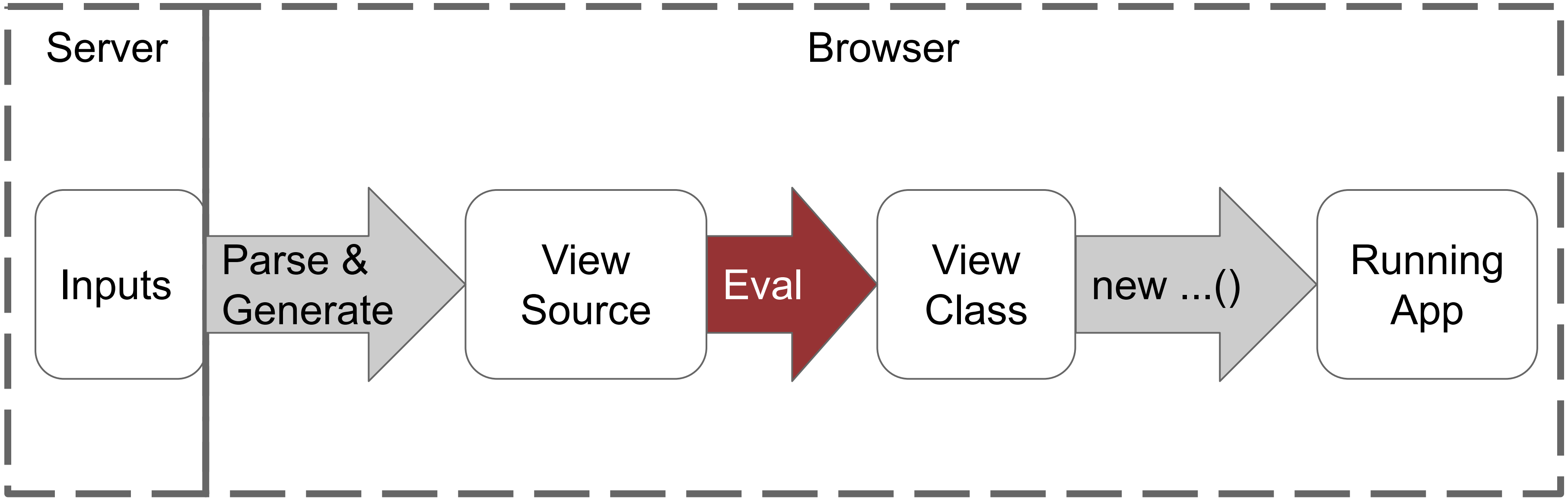
JIT Compilation - Overview



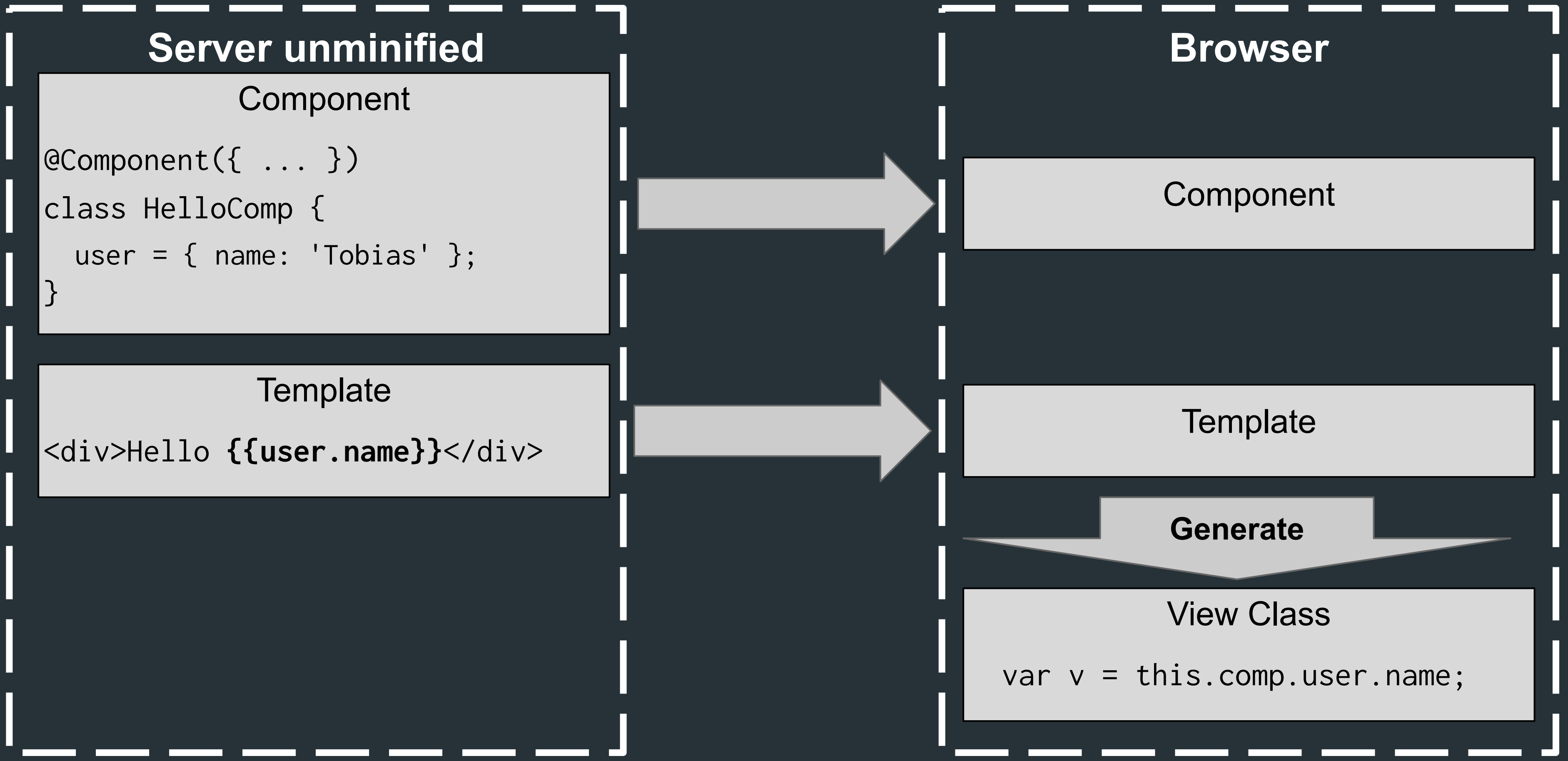
JIT Compilation - Overview



JIT Compilation - Overview



JIT Compilation - Enhanced Minification

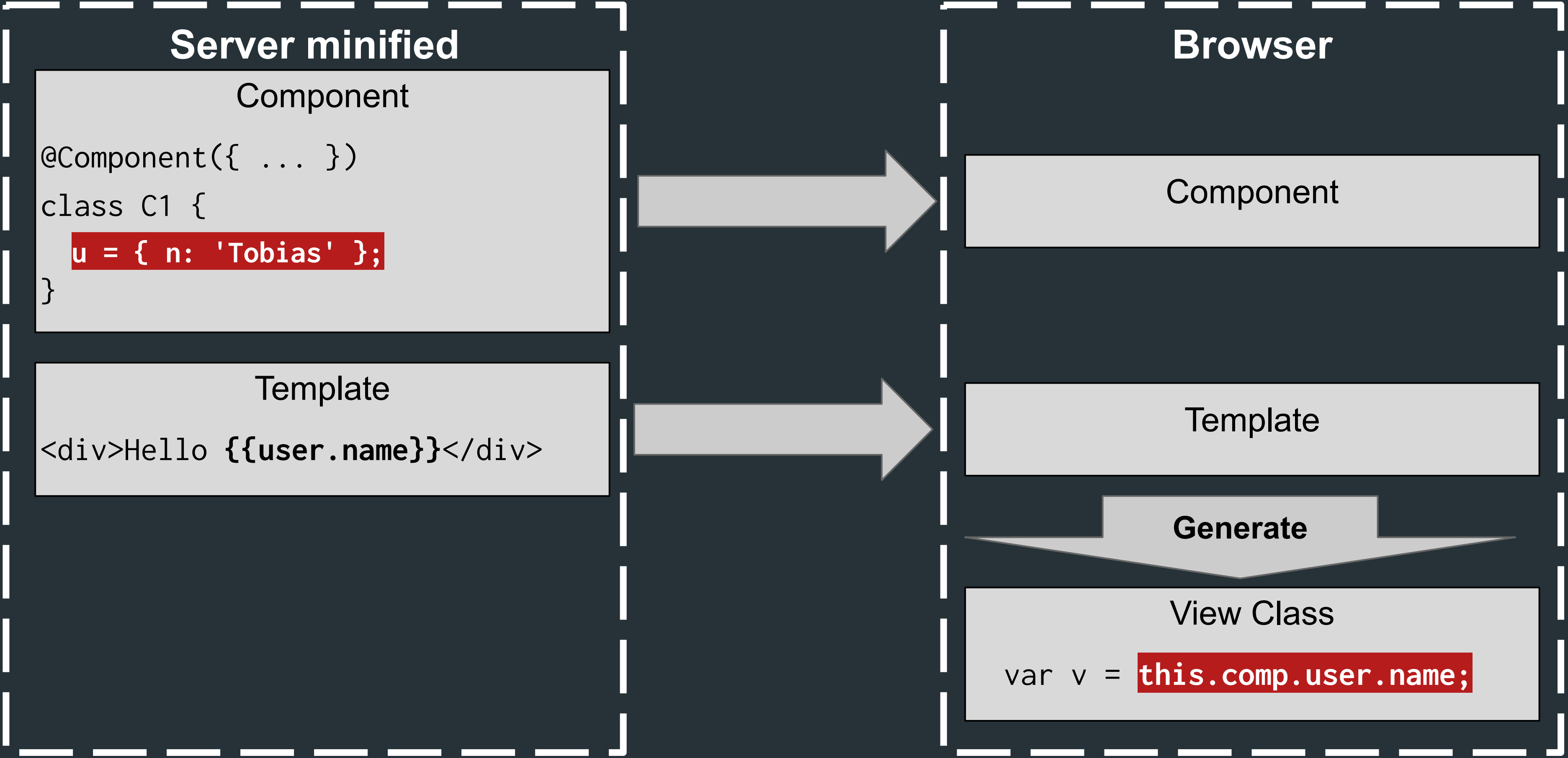


```
@Component({ ... })
class HelloComp {
  user = { name: 'Tobias' };
}
```

```
<div>Hello {{user.name}}</div>
```

```
var v = this.comp.user.name;
```

JIT Compilation - Enhanced Minification



Server minified

Component

```
@Component({ ... })
class C1 {
  u = { n: 'Tobias' };
}
```

Template

```
<div>Hello {{user.name}}</div>
```

Browser

Component

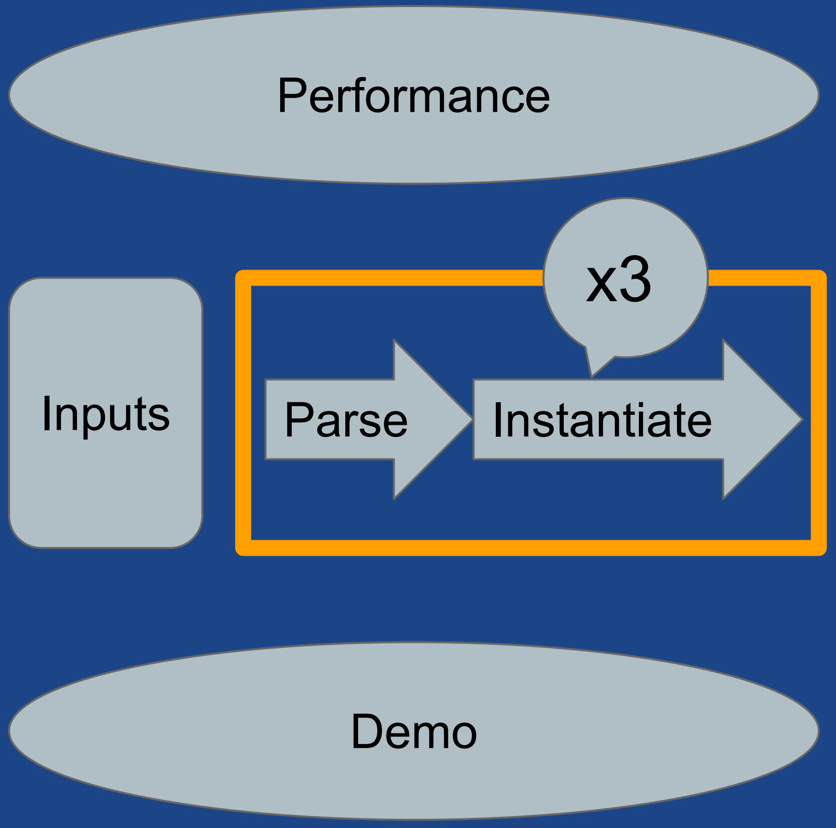
Template

Generate

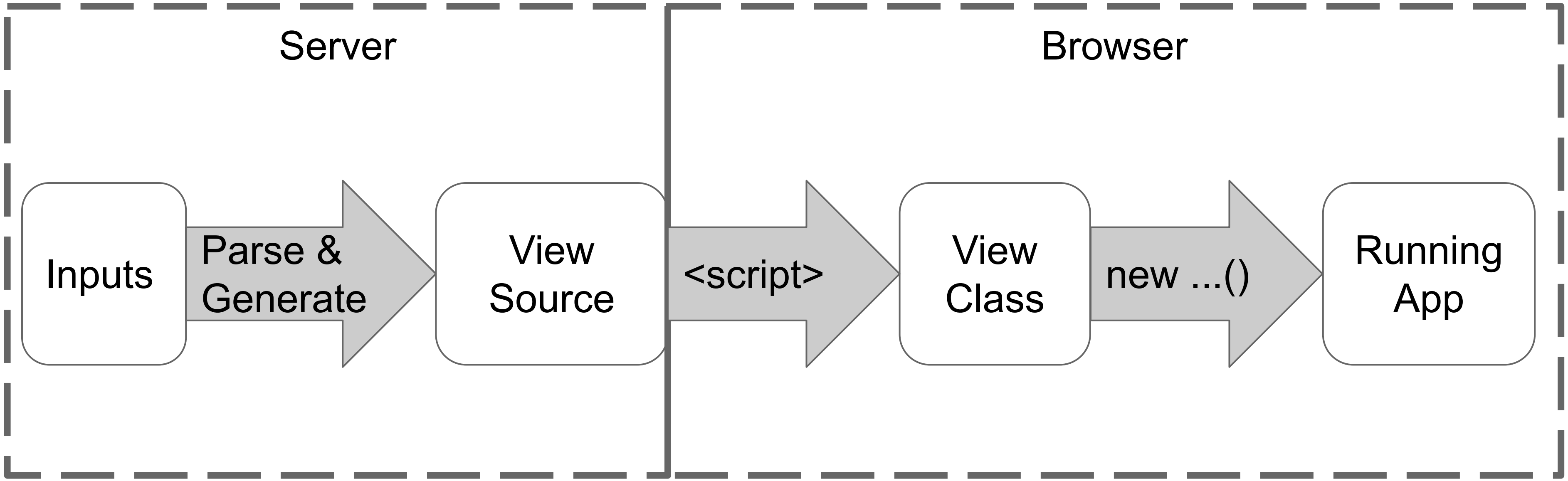
View Class

```
var v = this.comp.user.name;
```

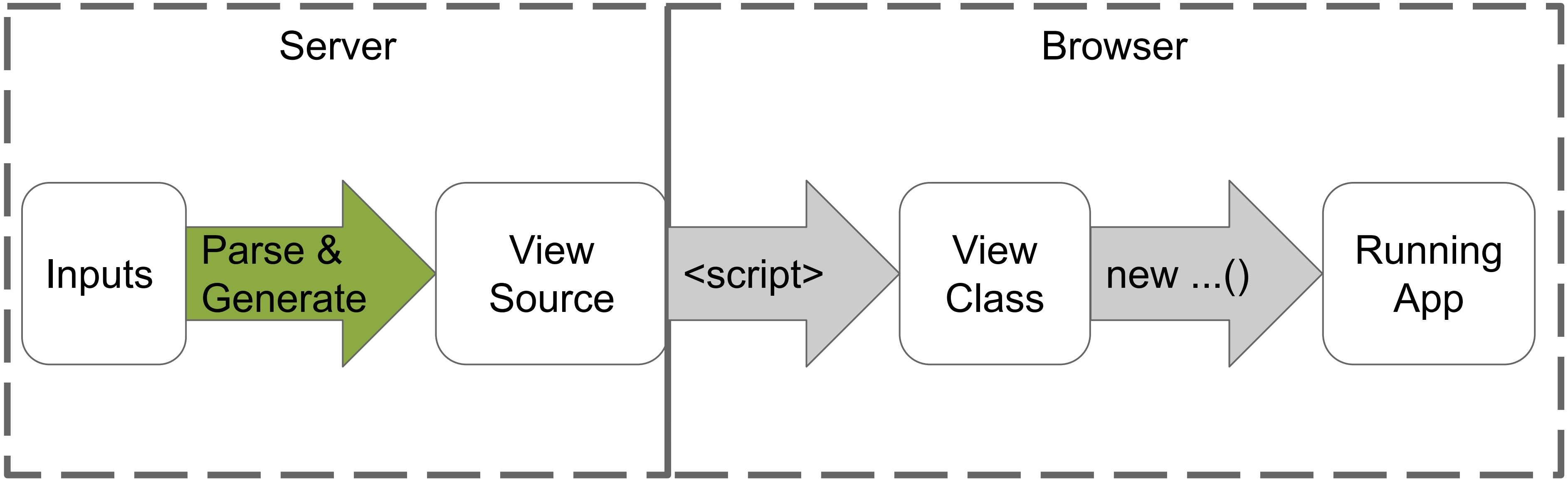
AOT Compilation



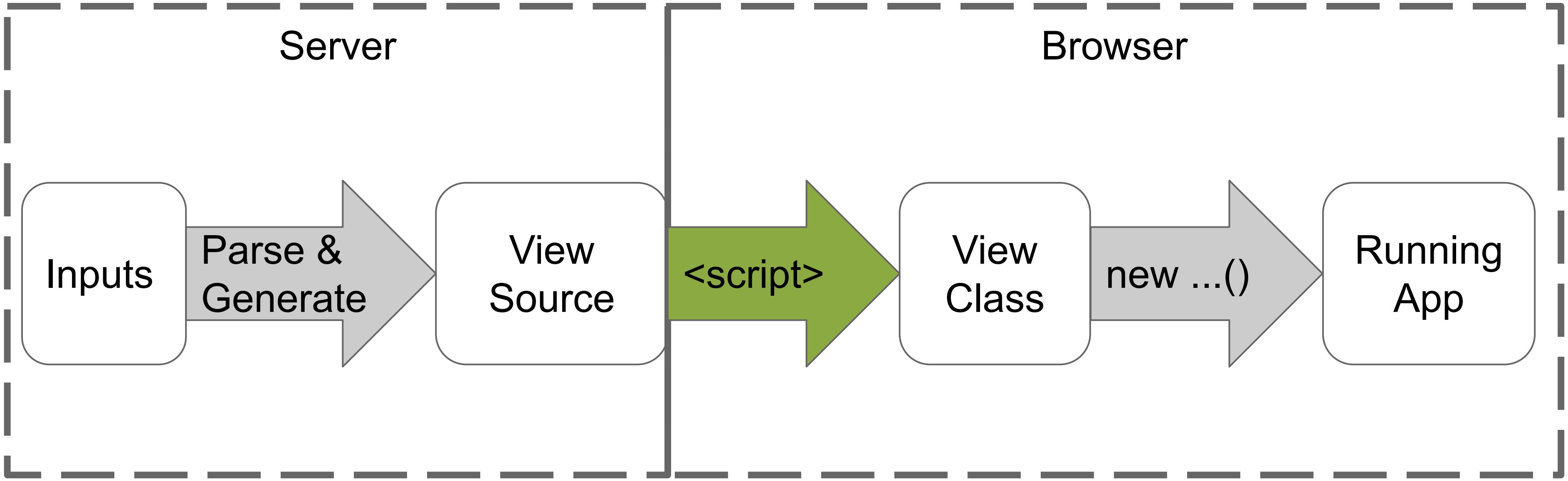
JIT Compilation - Overview



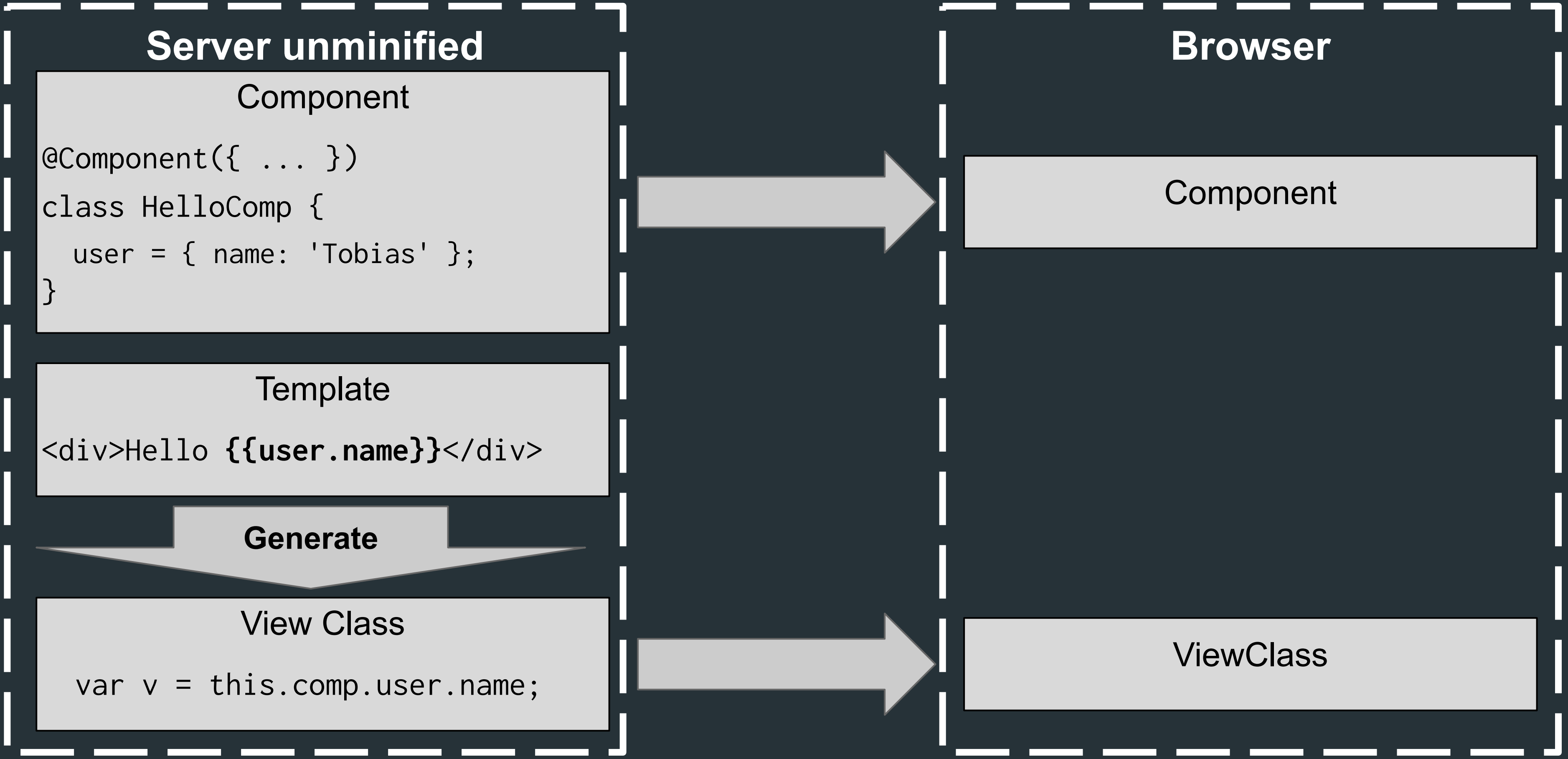
JIT Compilation - Overview



JIT Compilation - Overview



AOT Compilation - Enhanced Minification

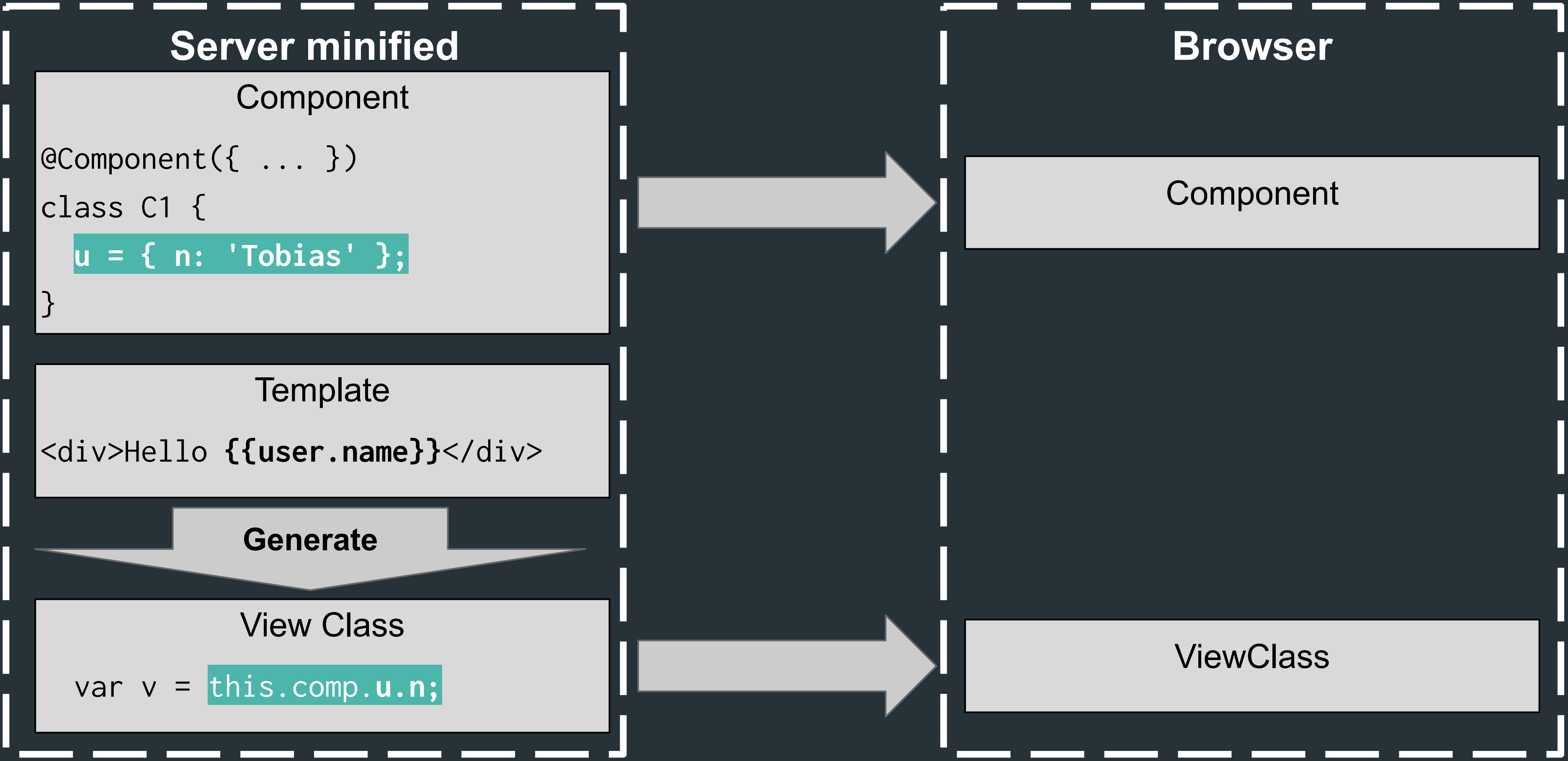


```
@Component({ ... })
class HelloComp {
  user = { name: 'Tobias' };
}
```

```
<div>Hello {{user.name}}</div>
```

```
  var v = this.comp.user.name;
```

AOT Compilation - Enhanced Minification

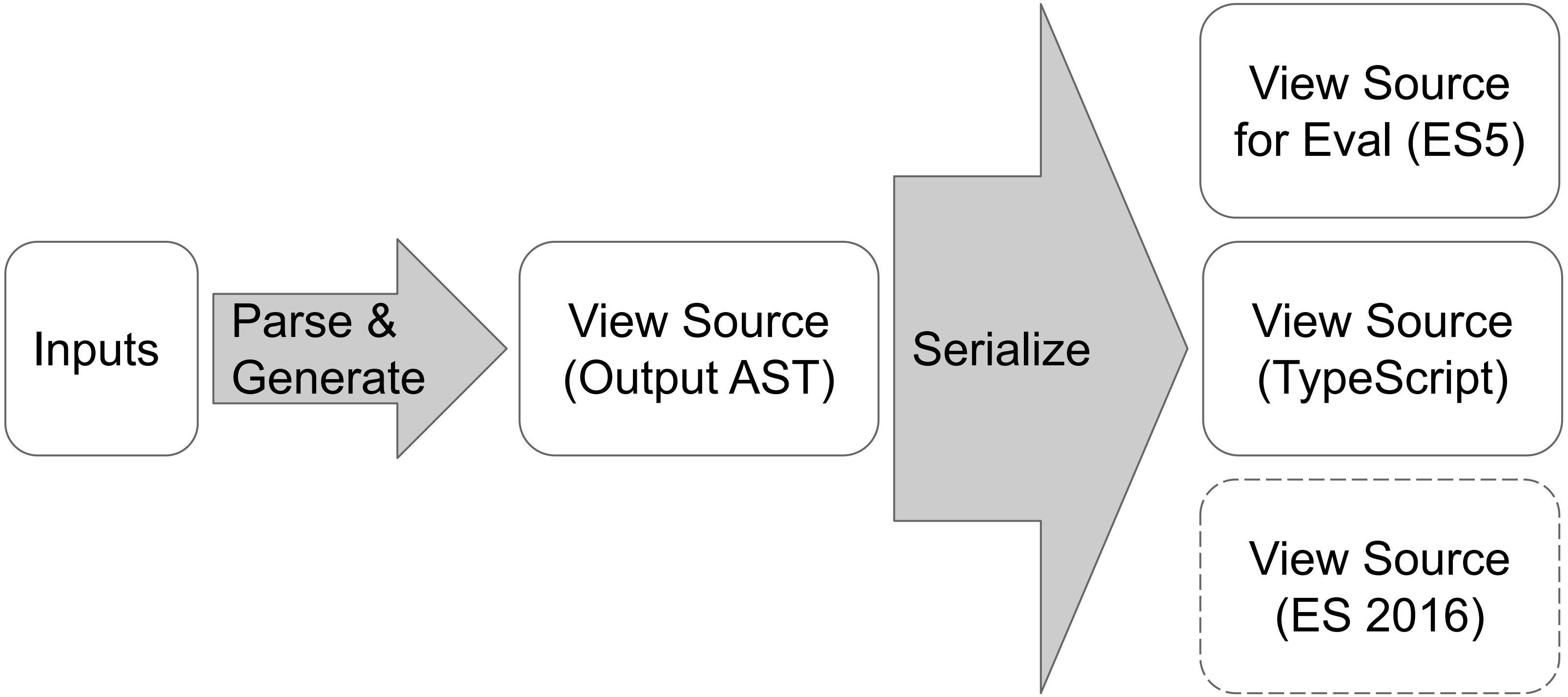


```
@Component({ ... })
class C1 {
  u = { n: 'Tobias' };
}
```

```
<div>Hello {{user.name}}</div>
```

```
  var v = this.comp.u.n;
```

AOT compilation - TypeScript Output



AOT Compilation - TypeScript Output

```
{ kind: 'declareVar', name: 'el',
  type: { kind: 'ambient', name: 'HTMLElement' },
  value: {
    kind: 'invokeMethod',
    receiver: { kind: 'readVar', name: 'document' },
    name: 'createElement',
    params: { kind: 'literal', value: 'div' }
  }
}
```

Generated ES5 code

```
var el = document
  .createElement('div');
```

Generated TypeScript code

```
var el: HTMLElement = document
  .createElement('div');
```

AOT Compilation - TypeScript Output

```
{ kind: 'declareVar', name: 'el',
  type: { kind: 'ambient', name: 'HTMLElement' },
  value: {
    kind: 'invokeMethod',
    receiver: { kind: 'readVar', name: 'document' },
    name: 'createElement',
    params: { kind: 'literal', value: 'div' }
  }
}
```

Generated ES5 code

```
var el = document
  .createElement('div');
```

Generated TypeScript code

```
var el: HTMLElement = document
  .createElement('div');
```

AOT Compilation - TypeScript Output

```
{ kind: 'declareVar', name: 'el',
  type: { kind: 'ambient', name: 'HTMLElement' },
  value: {
    kind: 'invokeMethod',
    receiver: { kind: 'readVar', name: 'document' },
    name: 'createElement',
    params: { kind: 'literal', value: 'div' }
  }
}
```

Generated ES5 code

```
var el = document
  .createElement('div');
```

Generated TypeScript code

```
var el: HTMLElement = document
  .createElement('div');
```

AOT Compilation - Metadata Collection

```
var supportsCookies =
  document.cookie;

@Directive({
  selector: '[someDir]',
})
class SomeDir {
  // use supportsCookies...
}
```

AOT Compilation - Metadata Collection

```
var supportsCookies =
  document.cookie;

function SomeDir() {
  // use supportsCookies...
}
SomeDir.annotations = [{
  selector: '[someDir]',
}];
```

AOT Compilation - Metadata collection

```
{ "SomeDir": {
    "decorators": [{
      "expression": { "module": "@angular/core",
                      "name": "Directive" },
      "arguments": [{ "selector": "[someDir]" }]
    }]
} }
```

Source

```
var supportsCookies =
  document.cookie;

@Directive({
  selector: '[someDir]',
})
class SomeDir {
  // use supportsCookies...
}
```

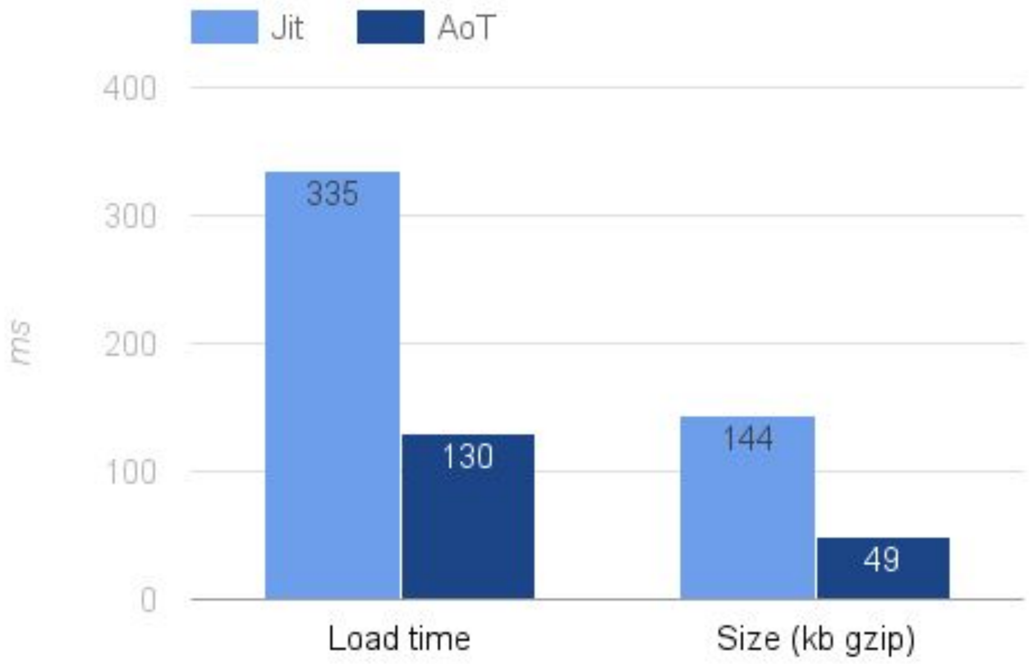
AOT Compilation - Metadata collection

```
{ "SomeDir": {
    "decorators": [{
      "expression": { "module": "@angular/core",
                      "name": "Directive" },
      "arguments": [{ "selector": "[someDir]" }]
    }]
} }
```

Source

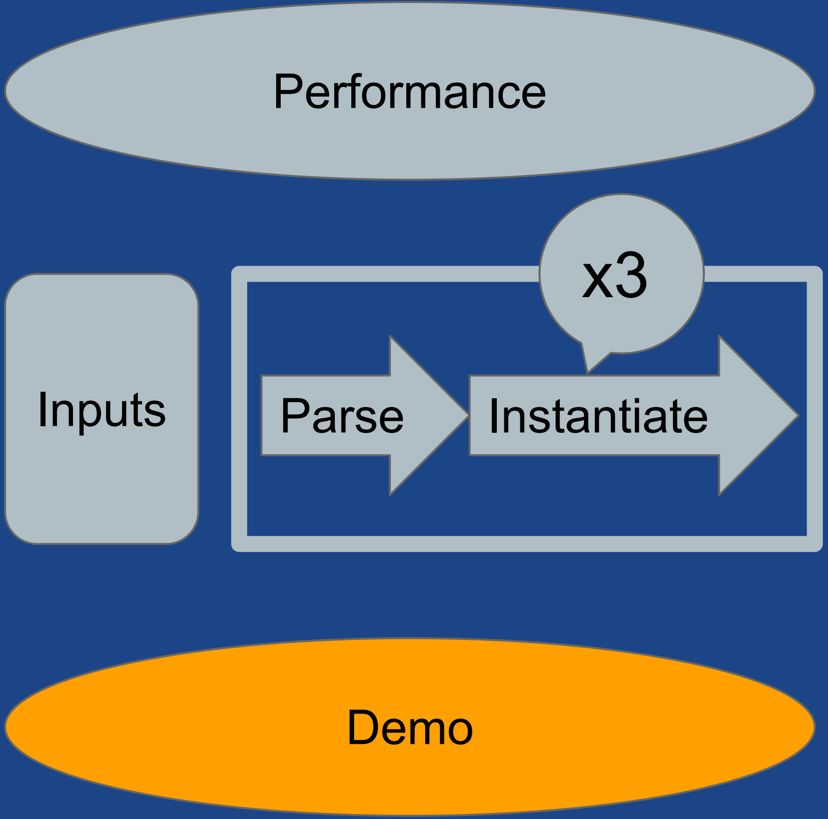
```
var supportsCookies =
  document.cookie;

@Directive({
  selector: '[someDir]',
})
class SomeDir {
  // use supportsCookies...
}
```

JIT vs AOT Compilation - Page Load Performance



Demo

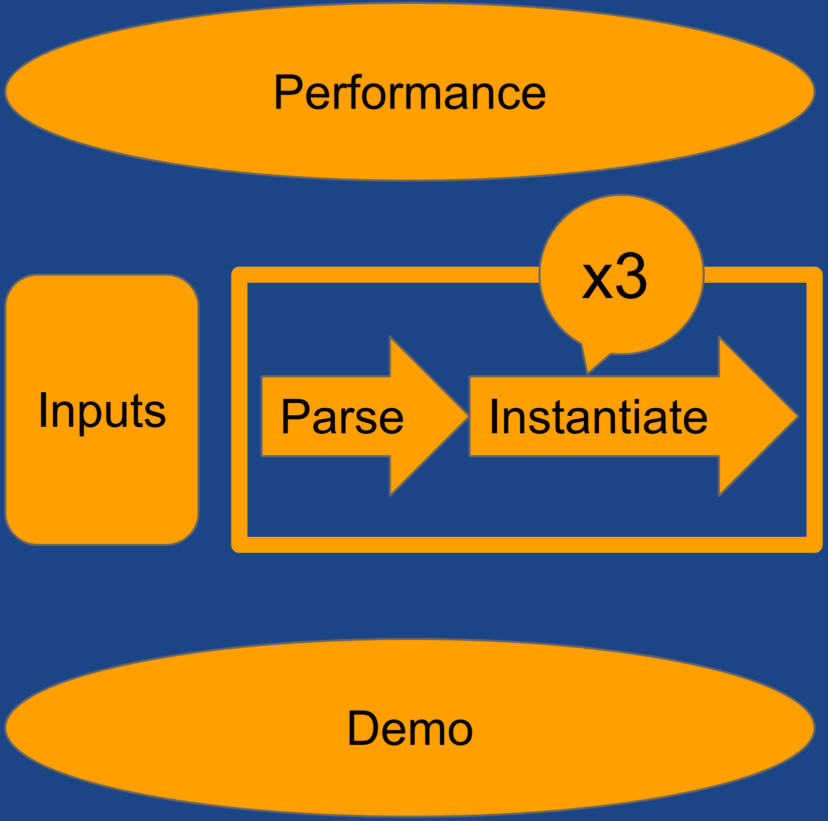
```
$ ngc -p <project>
```



What's next

Smaller: **10kb**

FTL: **F**aster **T**han base**L**ine



{ document.createElement() }

{ Fast Property Access / Hidden Classes [1] }

{ Codegen Classes }

{ Just in Time vs Ahead of Time Compile }

g.co/ng/ac16-compiler

[1] github.com/v8/v8/wiki/Design%20Elements