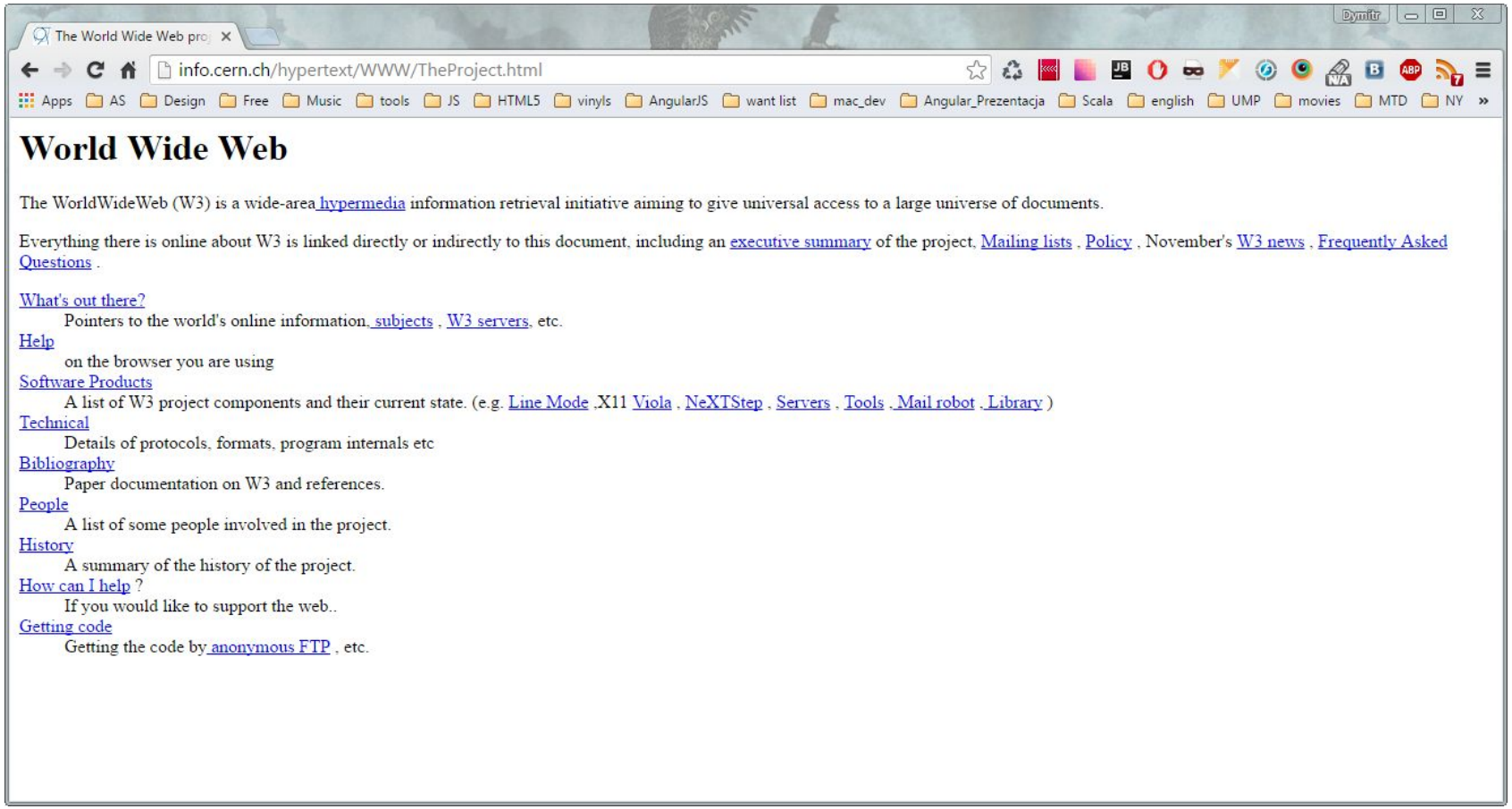
Front-end development w Scala.JS

https://github.com/AVSystem/Wyklady_2016

Front-end development

Trzy podstawowe moduły:

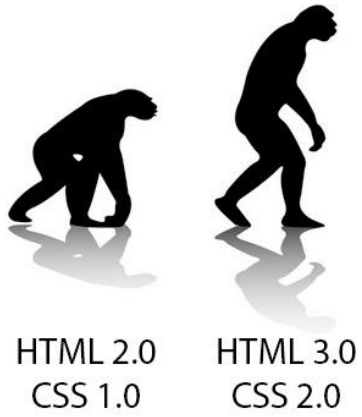
Pierwszy krok



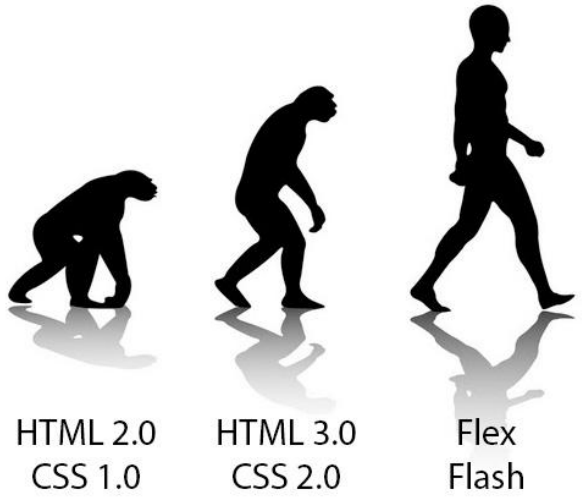
Ewolucja

HTML 2.0
CSS 1.0

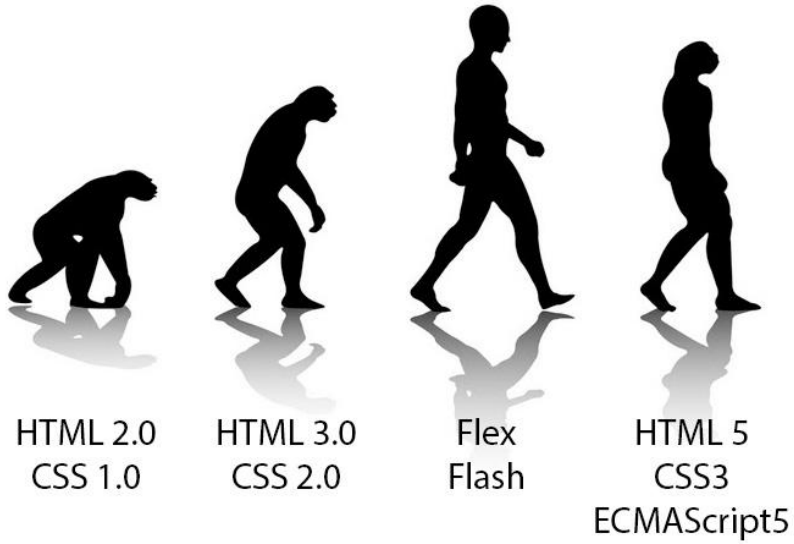
Ewolucja



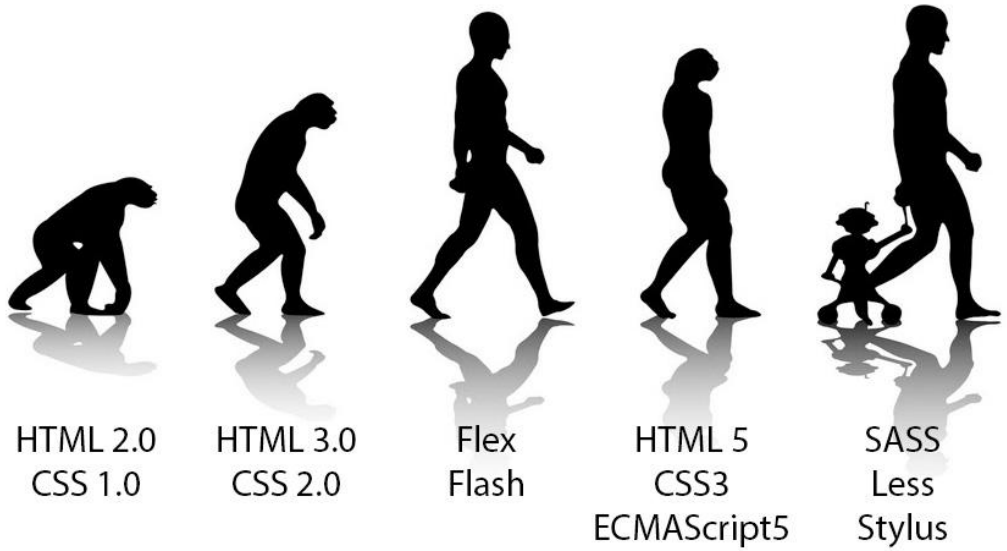
Ewolucja



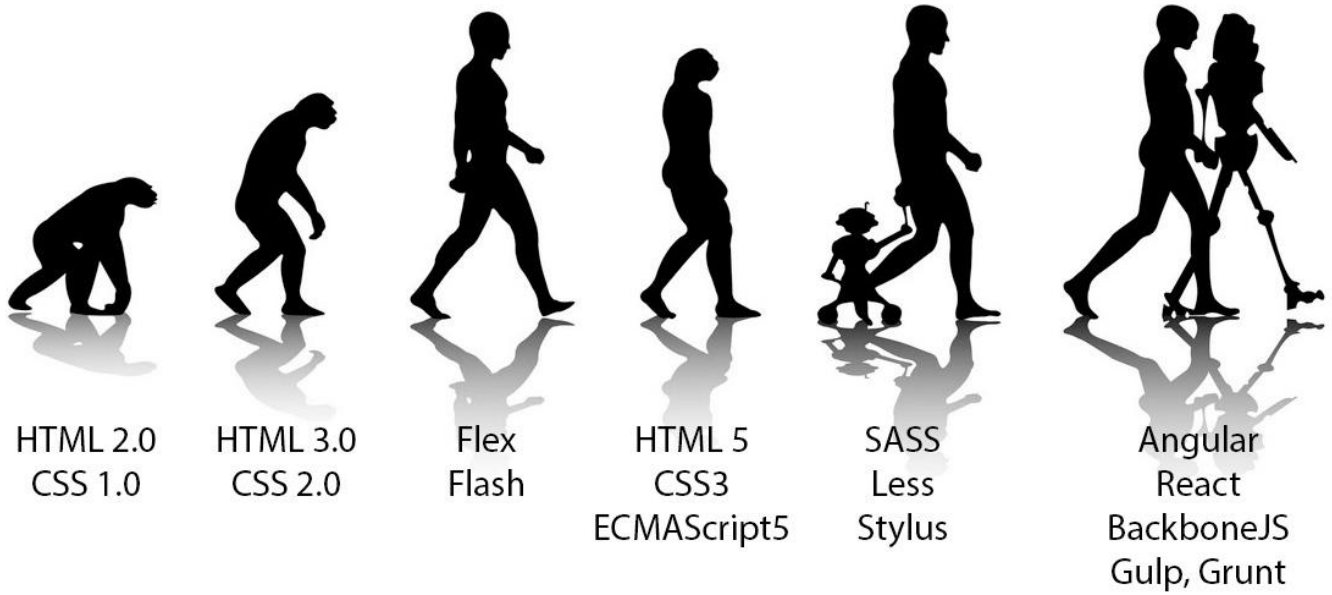
Ewolucja


HTML 2.0
CSS 1.0
HTML 3.0
CSS 2.0
Flex
Flash
HTML 5
CSS3
ECMAScript5

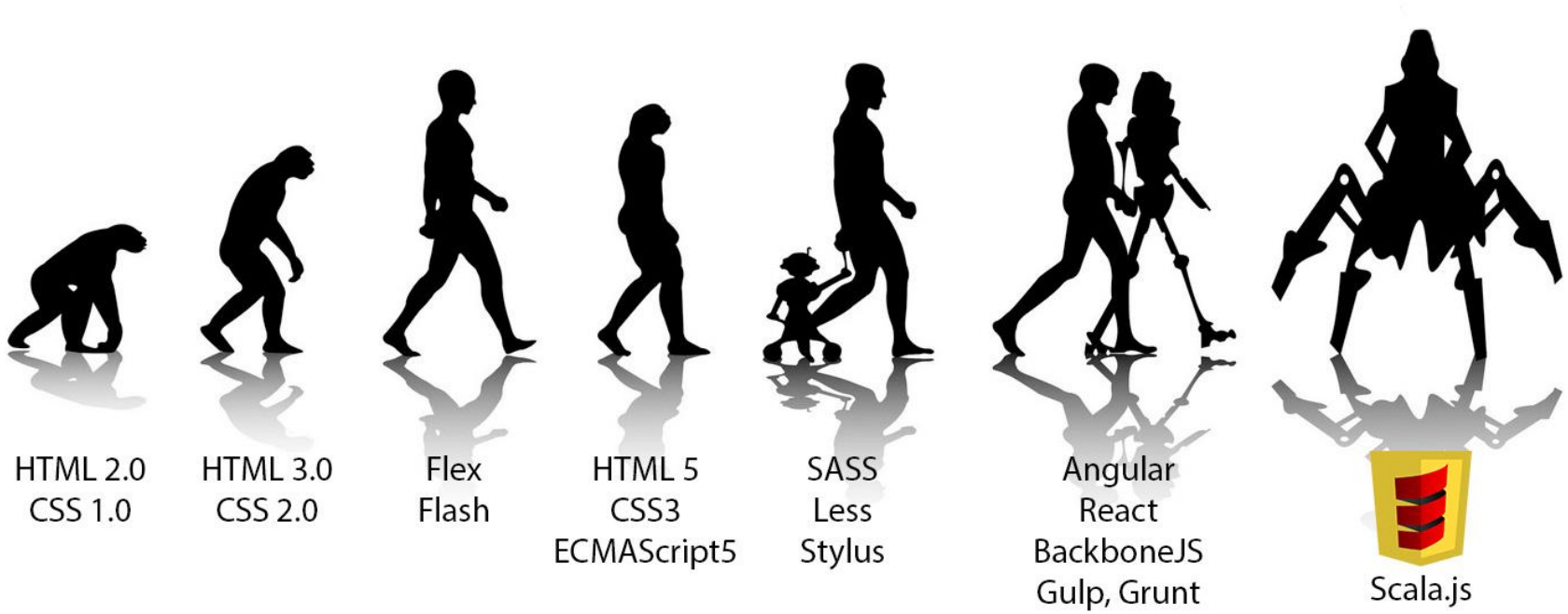
Ewolucja


HTML 2.0
CSS 1.0
HTML 3.0
CSS 2.0
Flex
Flash
HTML 5
CSS3
ECMAScript5
SASS
Less
Stylus

Ewolucja

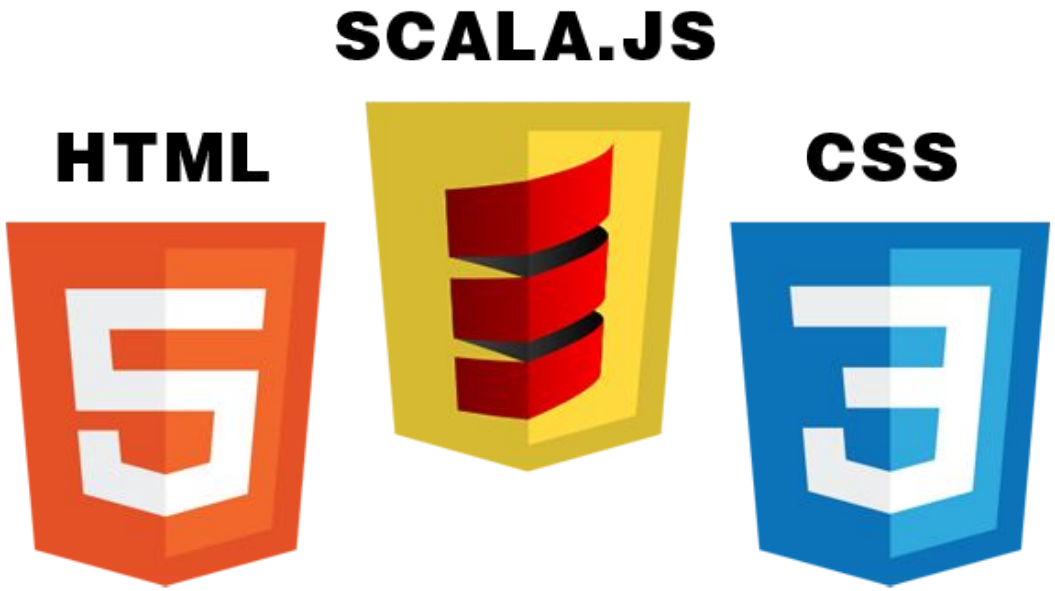

HTML 2.0
CSS 1.0
HTML 3.0
CSS 2.0
Flex
Flash
HTML 5
CSS3
ECMAScript5
SASS
Less
Stylus
Angular
React
BackboneJS
Gulp, Grunt

Ewolucja


HTML 2.0
CSS 1.0
HTML 3.0
CSS 2.0
Flex
Flash
HTML 5
CSS3
ECMAScript5
SASS
Less
Stylus
Angular
React
BackboneJS
Gulp, Grunt
Scala.js

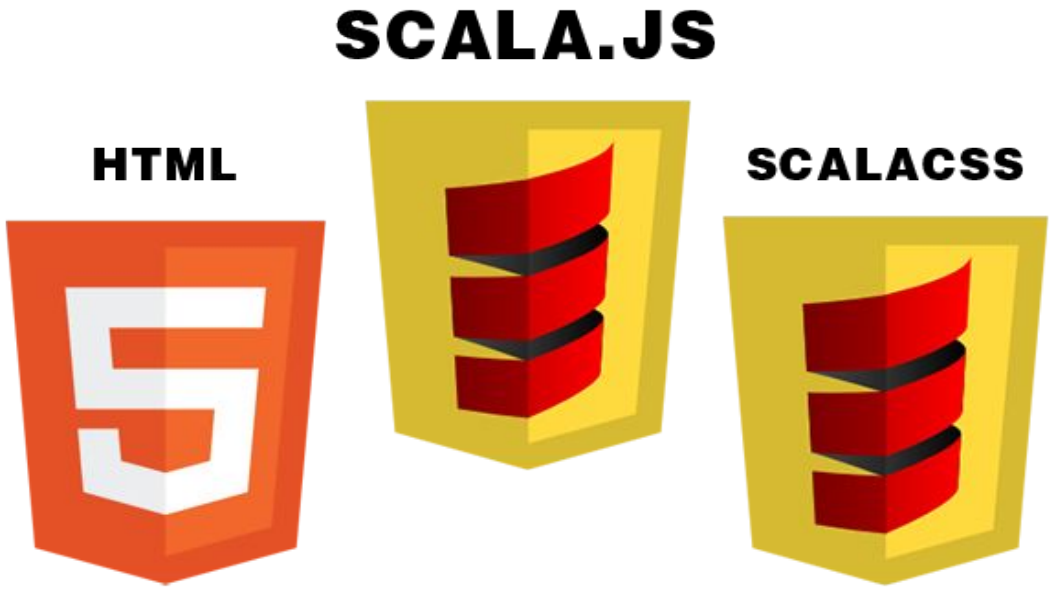
Front-end na nowo

Piszemy JavaScript nie używając Javasctipt



Front-end na nowo

CSS bez CSSa?



ScalaCSS

- Type-safe

ScalaCSS

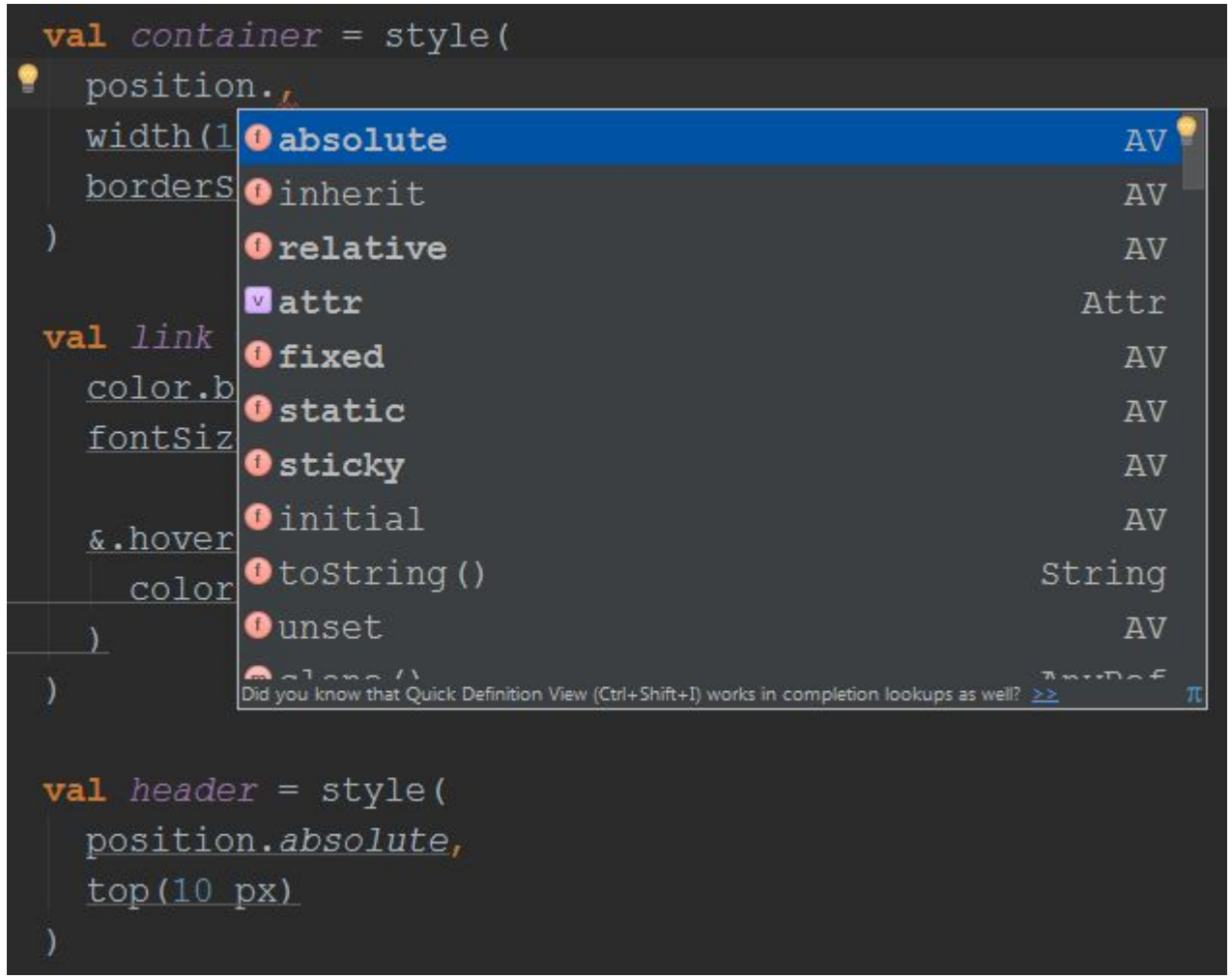
- Type-safe
- Przejrzystość

ScalaCSS

```scala
val container = style(
  position.relative,
  width(100 px),
  borderStyle.dotted
)

val link = style(
  color.black,
  fontSize(12 rem),

  &.hover (
    color.black
  )
)

val header = style(
  position.absolute,
  top(10 px)
)
```

```css
.container {
  position: relative;
  width: 100%;
  border-style: dotted;
}

.link {
  color: black;
  font-size: 12rem;

  &:hover {
    color: black;
  }
}

.header {
  position: absolute;
  top: 10px;
}
```

ScalaCSS


val container = style(
position.,
width(1
borderS
)
val link
color.b
fontSiz
&.hover
color
)
)
val header = style(
position.absolute,
top(10 px)
)
absolute AV
inherit AV
relative AV
attr Attr
fixed AV
static AV
sticky AV
initial AV
toString() String
unset AV
Did you know that Quick Definition View (Ctrl+Shift+I) works in completion lookups as well? >>

ScalaCSS

- Type-safe
- Przejrzystość
- Dynamiczny CSS

ScalaCSS

AVSYSTEM

```
val container = style(
  position.relative,
  width(100 %%),
  borderStyle.dotted
)

val containerRelated = style(
  position.relative,
  width(DomUtils.elementWidth(container) px),
  borderStyle.dotted
)
```

ScalaCSS

- Type-safe
- Przejrzystość
- Dynamiczny CSS
- Możliwości większe niż dają same pre-procesory

ScalaCSS

Wbudowany autoprefixer

```
val flexBox = style(
  display.flex
)
```

```
.flexBox {
    display: -webkit-box;
    display: -webkit-flex;
    display: -ms-flexbox;
    display: flex;
}
```

ScalaCSS

Post-processory nie są potrzebne

```
.theme-1 .link {
    color: #ffff00;
}

.theme-2 .link {
    color: #ff0000;
}

.theme-1 .content {
    background-color: #ffff00;
}

.theme-2 .content {
    background-color: #ffff00;
}
```

ScalaCSS

Post-processory nie są potrzebne

```
object Config {
  lazy val colorTheme: String = document.body.getAttribute("data-theme")
}

object Colors {
  private val alphaColors: Map[String, ValueT[ValueT.Color]] = Map(
    "theme-1" -> c"ffff00",
    "theme-2" -> c"ff0000"
  )

  def alpha = alphaColors.get(Config.colorTheme).get
}

val link = style(
  color(Colors.alpha)
)
```

ScalaCSS

- Type-safe
- Przejrzystość
- Dynamiczny CSS
- Możliwości większe niż dają same pre-procesory
- Łatwy refactoring

ScalaCSS

- Type-safe
- Przejrzystość
- Dynamiczny CSS
- Możliwości większe niż dają same pre-procesory
- Łatwy refactoring
- Współdzielony kod

ScalaCSS

Współdzielony kod

```
object Sizes {
  val headerHeight = 100
}
```

```
val header = style(
  position.relative,
  height(Sizes.headerHeight px)
)

val content = style(
  position.relative
)
```

```
val windowHeight = DomUtils.windowHeight
val contentHeight = windowHeight - Sizes.headerHeight

setElementHeight(DemoStyles.content.htmlClass, contentHeight)

def setElementHeight(className: String, height: Int):Unit = {
  //set element height
}
```

ScalaCSS

- Type-safe
- Przejrzystość
- Dynamiczny CSS
- Możliwości większe niż dają same pre-procesory
- Łatwy refactoring
- Współdzielony kod
- Wykrywanie konfliktów w stylach

ScalaCSS

Wykrywanie konfliktów w stylach

```
val container = style(
  position.relative,
  width(100 %%),
  paddingLeft(10 px)
)

val content = style(
  container,
  paddingLeft(20 px)
)
```

ScalaCSS

Wykrywanie konfliktów w stylach

```
val container = style(
  position.relative,
  width(100 %%),
  paddingLeft(10 px)
)

val content = style(
  container,
  paddingLeft(20 px)
)(Compose.trust)
```

ScalaCSS

Mixins

```
@mixin breakpoint($point) {
  @if $point == large {
    @media (min-width: 64.375em) { @content; }
  }
  @else if $point == medium {
    @media (min-width: 50em) { @content; }
  }
  @else if $point == small {
    @media (min-width: 37.5em)  { @content; }
  }
}

.page-wrap {
  width: 75%;
  @include breakpoint(large) { width: 60%; }
  @include breakpoint(medium) { width: 80%; }
  @include breakpoint(small) { width: 95%; }
}
```

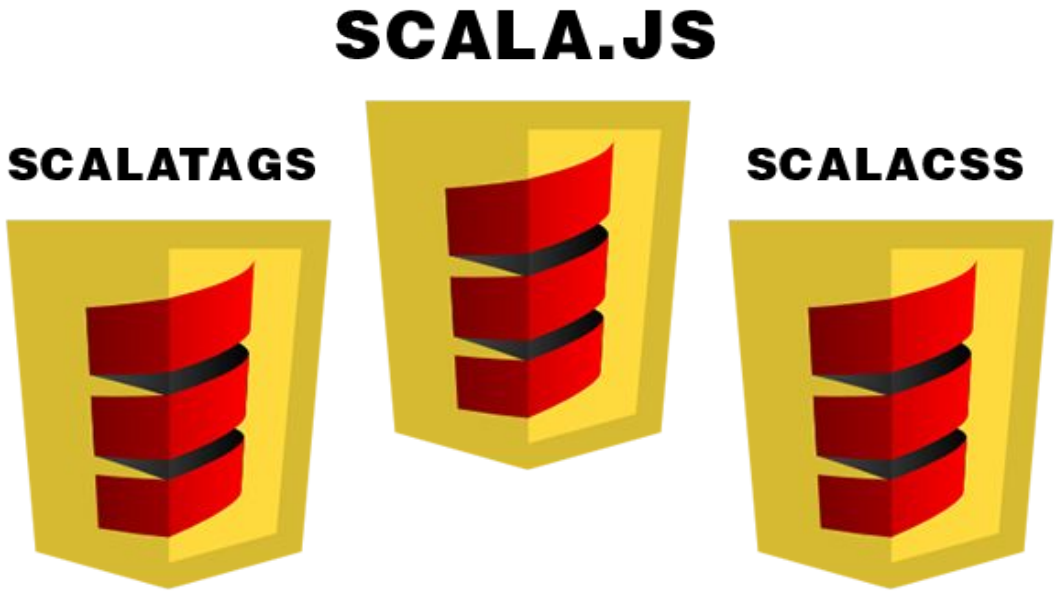
ScalaCSS

Mixins

```
trait BreakpointSize
object MediaQueries extends StyleSheet.Inline {

  case object Large extends BreakpointSize
  case object Small extends BreakpointSize

  def breakpoint(size: BreakpointSize)(properties: StyleA) = size match {
    case Large => style(
      media.screen.minWidth(64.375 em) (properties)
    )
    case Small => style(
      media.screen.minWidth(37 em) (properties)
    )
  }
}

val container = style(
  width(100 %%),
  MediaQueries.breakpoint(MediaQueries.Large)(style(
    width(60 px)
  ))
)
```

Front-end na nowo

HTML bez HTMLa?



ScalaTags

- Type-safe

ScalaTags

- Type-safe
- Czytelny kod

ScalaTags

Czytelny kod

```
val htmlTemplate = html(
  head(
    script(src := "some_script")
  ),
  body(
    h1("This is my title"),
    div(
      p("This is my first paragraph"),
      p("This is my second paragraph")
    )
  )
)
```

ScalaTags

Czytelny kod

```
<html>
    <head>
        <script src="some_script"> </script>
    </head>
    <body>
        <h1>This is my title</h1>
        <div>
            <p>This is my first paragraph</p>
            <p>This is my second paragraph</p>
        </div>
    </body>
</html>
```

ScalaTags

- Type-safe
- Czytelny kod
- Wykrywanie błedów

ScalaTags

Wykrywanie błędów

```
val htmlTemplate = html(
  head(
    script(src := "some_script")
  ),
  boby(
    is my title"),
    div(
      p("This is my first paragraph"),
      p("This is my second paragraph")
    )
  )
)
```

Cannot resolve symbol boby

ScalaTags

- Type-safe
- Czytelny kod
- Wykrywanie błedów
- Potężne narzędzie do szablonowania

ScalaTags

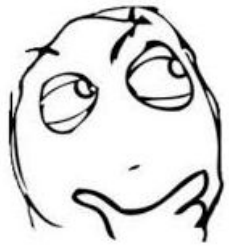
Szablony w BackboneJS

```
<ul>
    <% _.each(frameworks, function(framework) { %>
    <li title="<%- framework.description %>">
        <%- framework.name %>
    </li>
    <% }); %>
</ul>
```

ScalaTags

Szablony w EmberJS

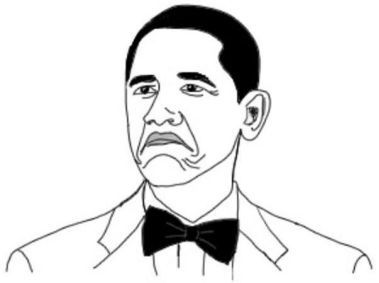
```
<ul>
    {{#each frameworks}}
    <li {{bind-attr title=description}}>
        {{name}}
    </li>
    {{/each}}
</ul>
```



ScalaTags

Szablony w AngularJS

```
<ul>
    <li ng-repeat="framework in frameworks"
        title="{{framework.description}}">
        {{framework.name}}
    </li>
</ul>
```



ScalaTags

Szablony w ScalaTags

```
ul(
  frameworks.map(framework =>
    li(title := framework.title)(framework.name)
  )
)
```

ScalaTags

Dowolne operacje bezpośrednio w szablonie

```
ul(
  frameworks
    .filter(_.name.contains("scala"))
    .map(f =>
      li(title := f.title.substring(1, 3))(
        f.name.toUpperCase()
      )
  )
)
```

ScalaTags

Ponowne użycie i łączenie szablonów

```
val headerTemplate = header("Header content")
val footerTemplate = footer("Footer content")

val pageTemplate = body(
  headerTemplate,
  div("Page content"),
  footerTemplate
)
```

ScalaTags

Dziedziczenie

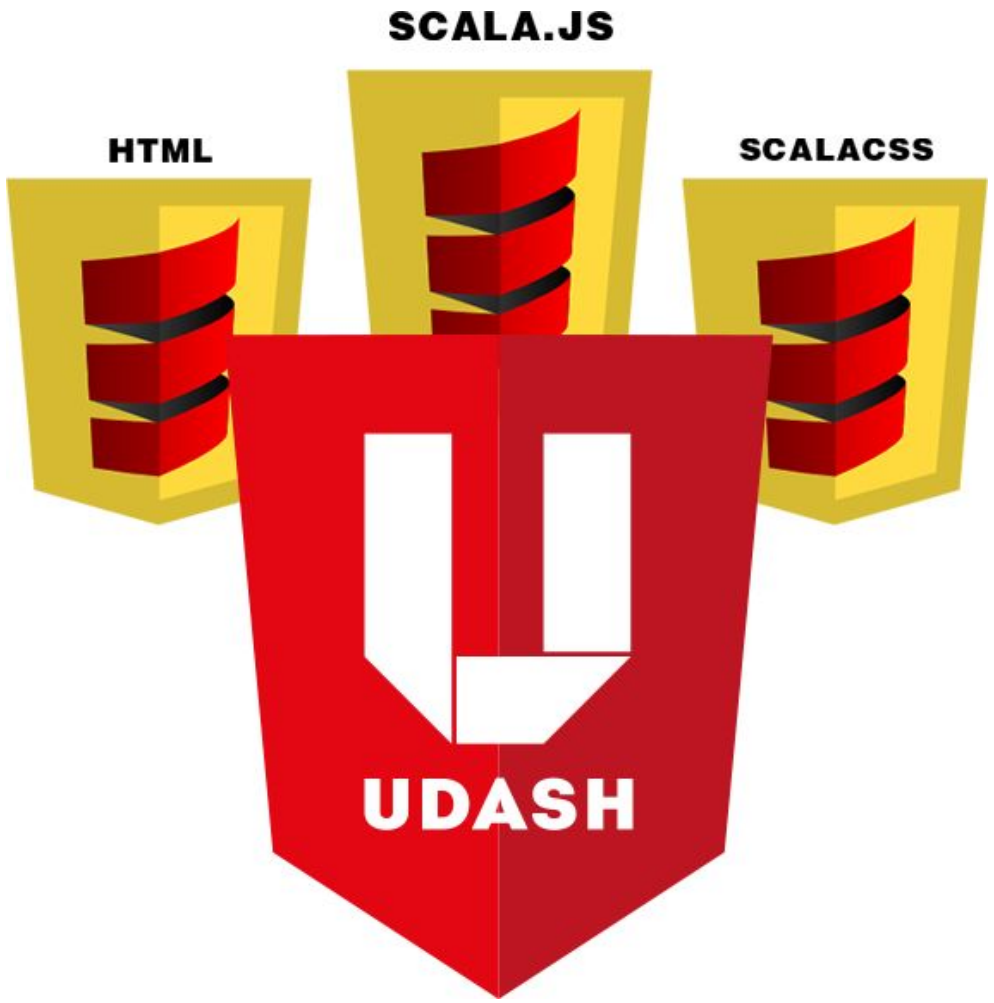
```
class Parent {
  def headTemplate = head("Head template")
  def bodyTemplate = body("Body content")

  def render = html(
    headTemplate,
    bodyTemplate
  )
}

object Child extends Parent {
  override def headTemplate = head("Some other head template")
}

val htmlPage = Child.render
```

ScalaTags + ScalaCSS

```
object DemoStyles extends StyleSheet.Inline {
  import dsl._

  val header = style(
    height(50 px)
  )

  val footer = style(
    position.fixed,
    bottom(`0`)
  )

  val content = style(
    position.relative,
    textAlign.center
  )
}
```

```
val headerTemplate = header(DemoStyles.header, "Header template")
val footerTemplate = footer(DemoStyles.footer, "Footer content")

val pageTemplate = body(
  headerTemplate,
  div(DemoStyles.content, "Page content"),
  footerTemplate
)
```

Udash Framework



Dziękujemy za uwagę

https://github.com/AVSystem/Wyklady_2016