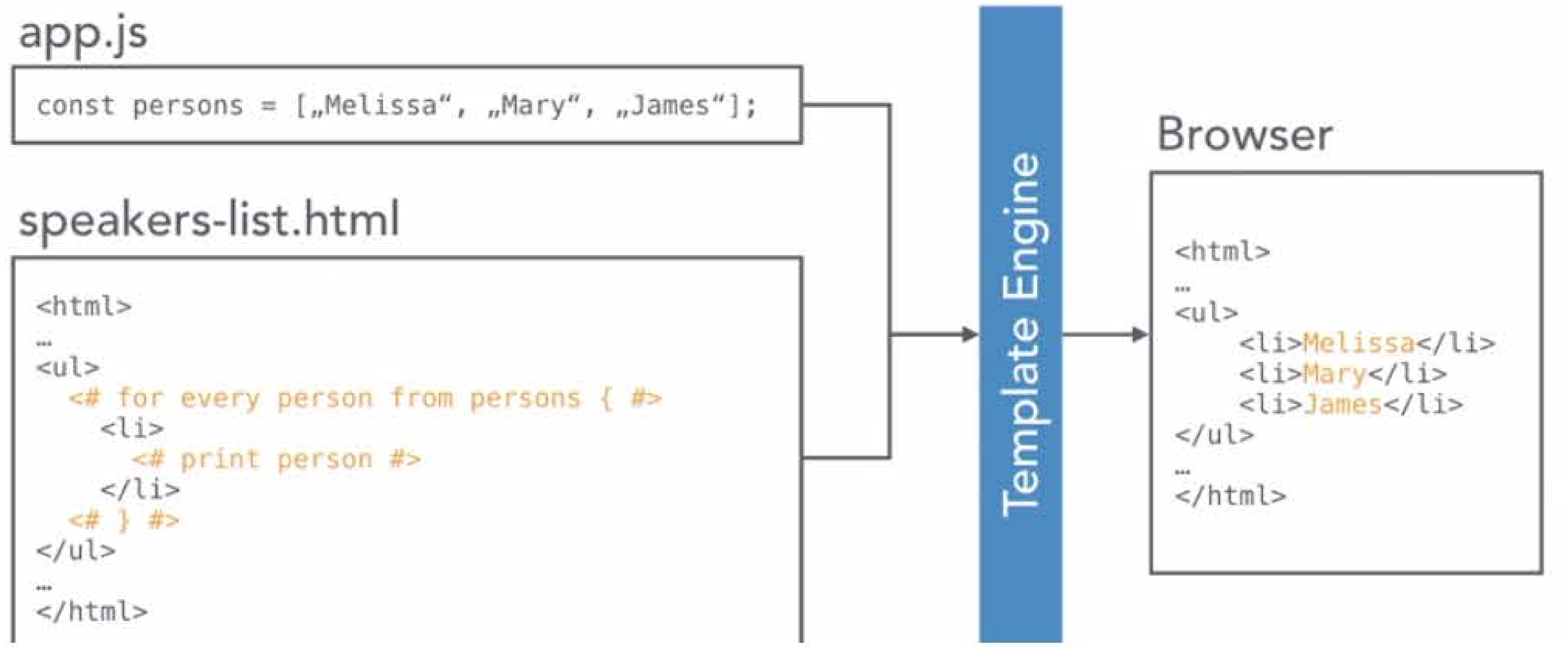
Template Engines and Express



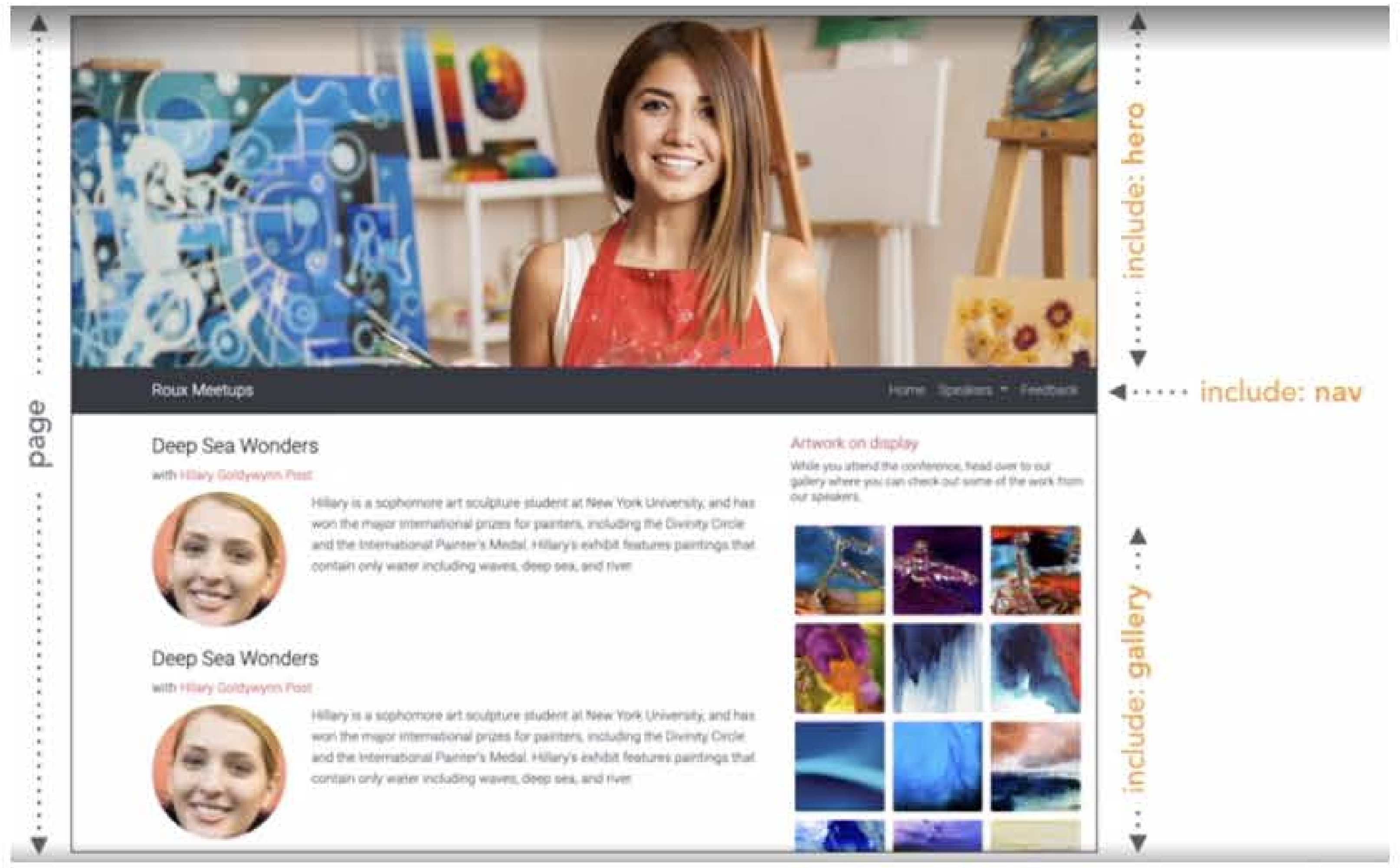
app.js

```
const persons = [„Melissa“, „Mary“, „James“];
```

speakers-list.html

```
<html>
…
<ul>
  <# for every person from persons { #>
    <li>
      <# print person #>
    </li>
  <# } #>
</ul>
…
</html>
```

```
<html>
…
<ul>
    <li>Melissa</li>
    <li>Mary</li>
    <li>James</li>
</ul>
…
</html>
```



Deep Sea Wonders

with Hillary Goldywynn Post

Hillary is a sophomore art sculpture student at New York University, and has won the major international prizes for painters, including the Divinity Circle and the International Painter's Medal. Hillary's exhibit features paintings that contain only water including waves, deep sea, and river.

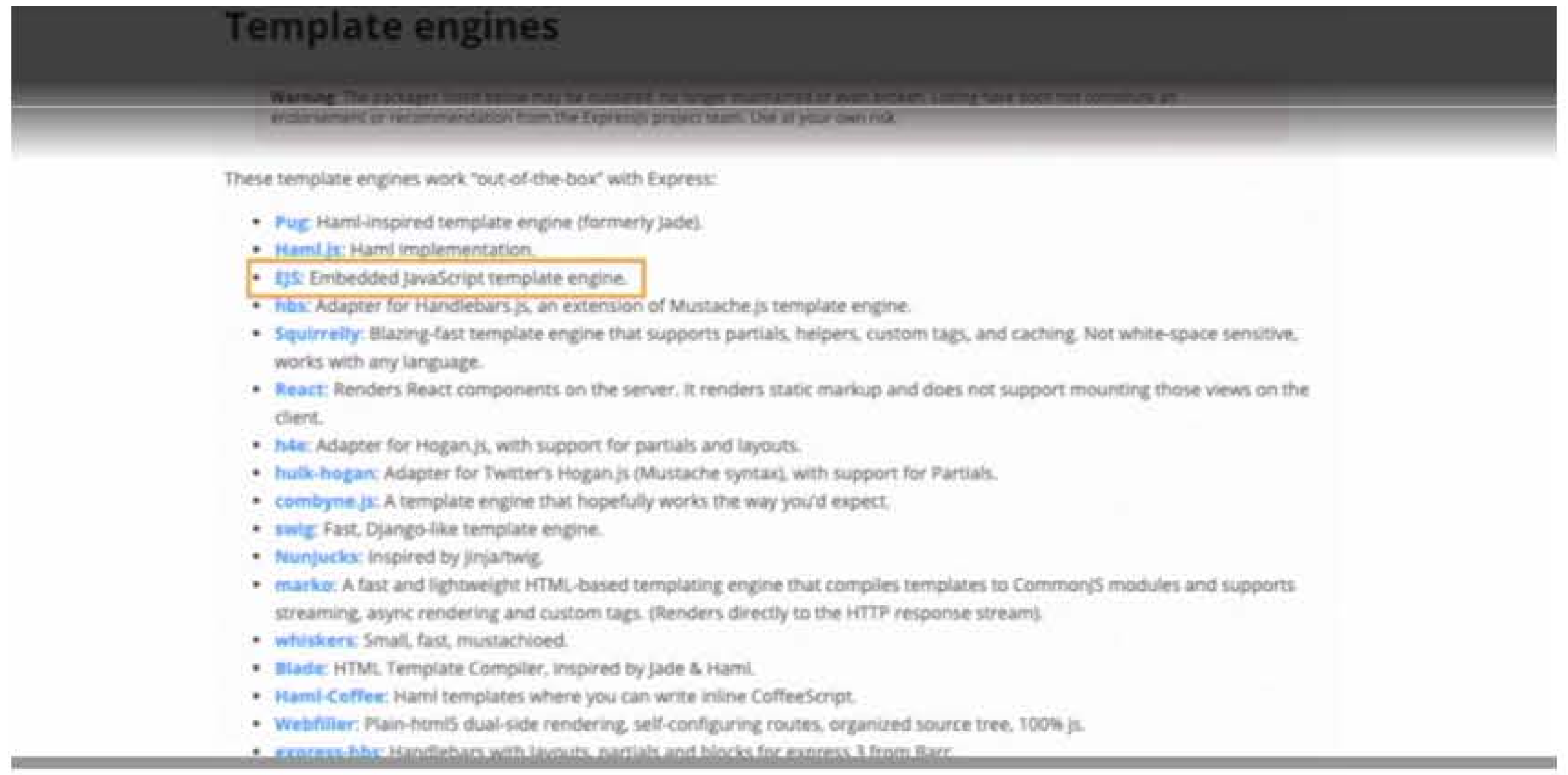
Deep Sea Wonders

with Hillary Goldywynn Post

Hillary is a sophomore art sculpture student at New York University, and has won the major international prizes for painters, including the Divinity Circle and the International Painter's Medal. Hillary's exhibit features paintings that contain only water including waves, deep sea, and river.

Artwork on display

While you attend the conference, head over to our gallery where you can check out some of the work from our speakers.



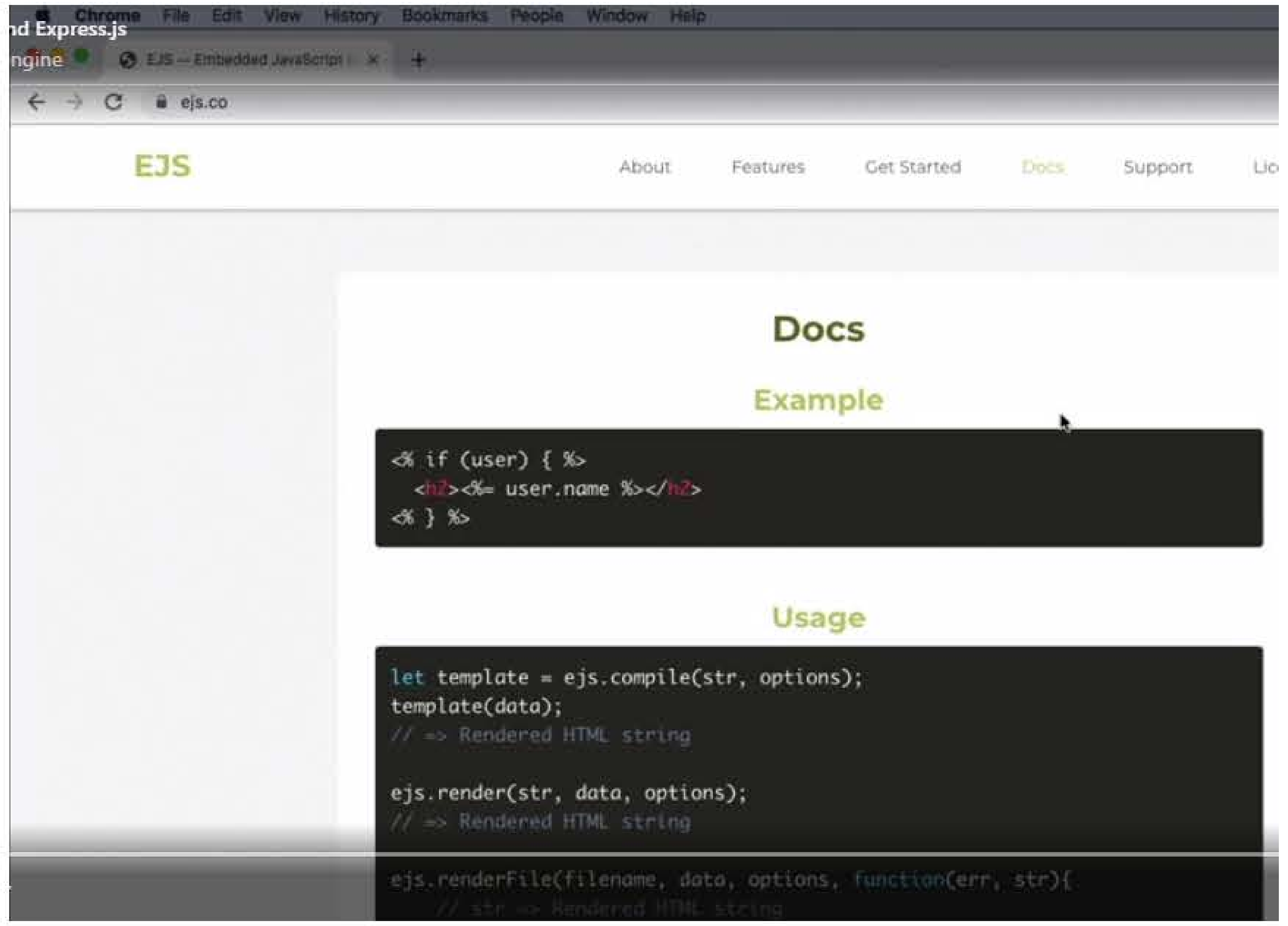
Template engines

Warning: The packages listed below may be outdated, no longer maintained or even broken. Listing here does not constitute an endorsement or recommendation from the Expressjs project team. Use at your own risk.

These template engines work "out-of-the-box" with Express:

- Pug: Haml-inspired template engine (formerly Jade).
- Haml.js: Haml implementation.
- EJS: Embedded JavaScript template engine.
- hbs: Adapter for Handlebars.js, an extension of Mustache.js template engine.
- Squirrelly: Blazing-fast template engine that supports partials, helpers, custom tags, and caching. Not white-space sensitive, works with any language.
- React: Renders React components on the server. It renders static markup and does not support mounting those views on the client.
- h4e: Adapter for Hogan.js, with support for partials and layouts.
- hulk-hogan: Adapter for Twitter's Hogan.js (Mustache syntax), with support for Partials.
- combyne.js: A template engine that hopefully works the way you'd expect.
- swig: Fast, Django-like template engine.
- Nunjucks: Inspired by jinja/twig.
- marko: A fast and lightweight HTML-based templating engine that compiles templates to CommonJS modules and supports streaming, async rendering and custom tags. (Renders directly to the HTTP response stream).
- whiskers: Small, fast, mustachioed.
- Blade: HTML Template Compiler, inspired by Jade & Haml.
- Haml-Coffee: Haml templates where you can write inline CoffeeScript.
- Webfiller: Plain-html5 dual-side rendering, self-configuring routes, organized source tree, 100% js.
- express-hbs: Handlebars with layouts, partials and blocks for express 3 from Barc.

Url: https://expressjs.com/en/resources/template-engines.html



Chrome File Edit View History Bookmarks People Window Help
nd Express.js
ngine
EJS -- Embedded JavaScript
ejs.co
EJS
About
Features
Get Started
Docs
Support
Lic

Docs

Example

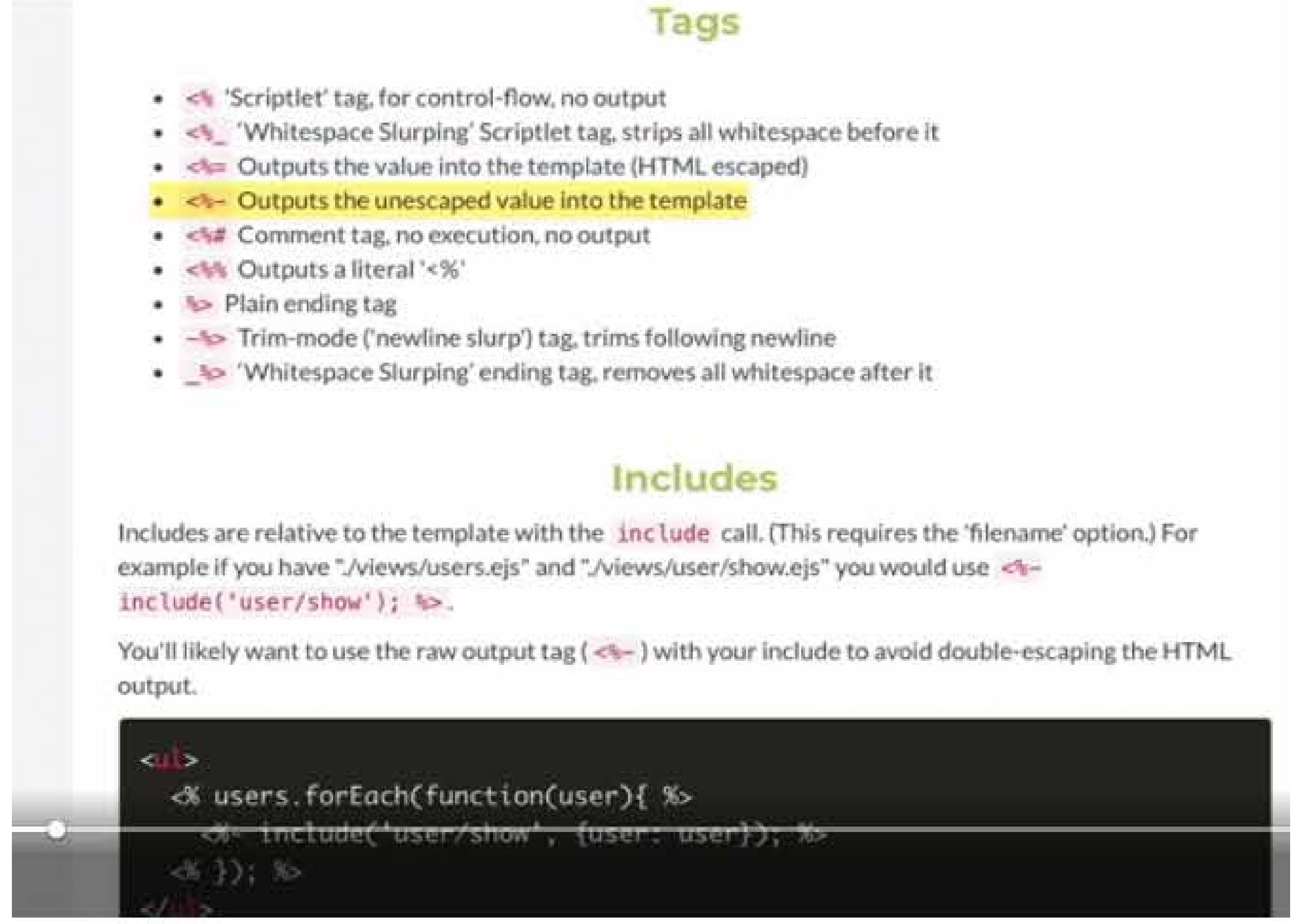
```
<% if (user) { %>
  <h2><%= user.name %></h2>
<% } %>
```

Usage

```
let template = ejs.compile(str, options);
template(data);
// => Rendered HTML string

ejs.render(str, data, options);
// => Rendered HTML string

ejs.renderFile(filename, data, options, function(err, str){
    // str => Rendered HTML string
```



Tags

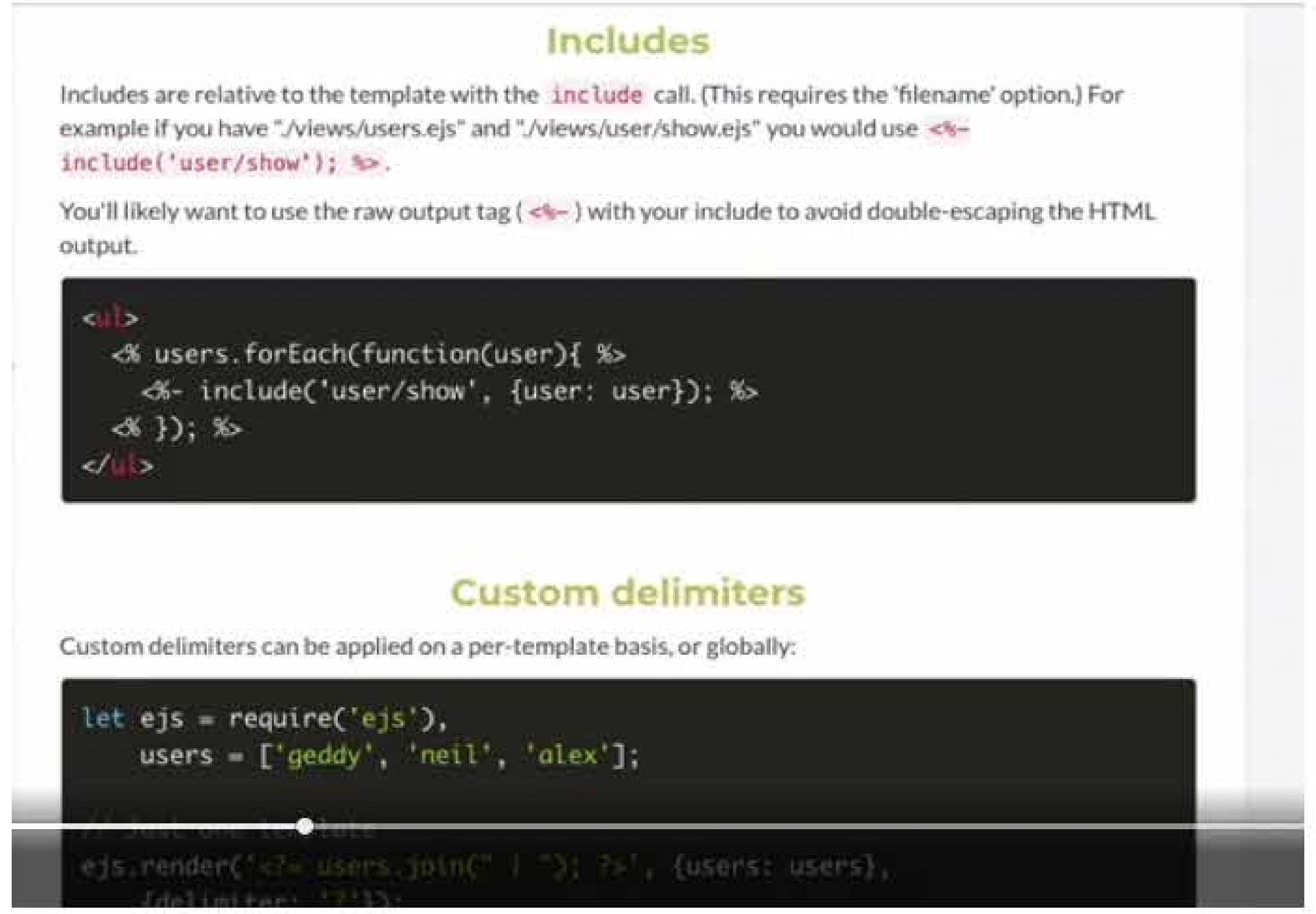
- `<%` 'Scriptlet' tag, for control-flow, no output
- `<%_` 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- `<%=` Outputs the value into the template (HTML escaped)
- `<%-` Outputs the unescaped value into the template
- `<%#` Comment tag, no execution, no output
- `<%%` Outputs a literal '<%'
- `%>` Plain ending tag
- `-%>` Trim-mode ('newline slurp') tag, trims following newline
- `_%>` 'Whitespace Slurping' ending tag, removes all whitespace after it

Includes

Includes are relative to the template with the `include` call. (This requires the 'filename' option.) For example if you have "./views/users.ejs" and "./views/user/show.ejs" you would use `<%- include('user/show'); %>`.

You'll likely want to use the raw output tag (`<%-`) with your include to avoid double-escaping the HTML output.

```
<ul>
  <% users.forEach(function(user){ %>
    <%- include('user/show', {user: user}); %>
  <% }); %>
</ul>
```



Includes

Includes are relative to the template with the `include` call. (This requires the 'filename' option.) For example if you have "./views/users.ejs" and "./views/user/show.ejs" you would use `<%- include('user/show'); %>`.

You'll likely want to use the raw output tag (`<%-`) with your include to avoid double-escaping the HTML output.

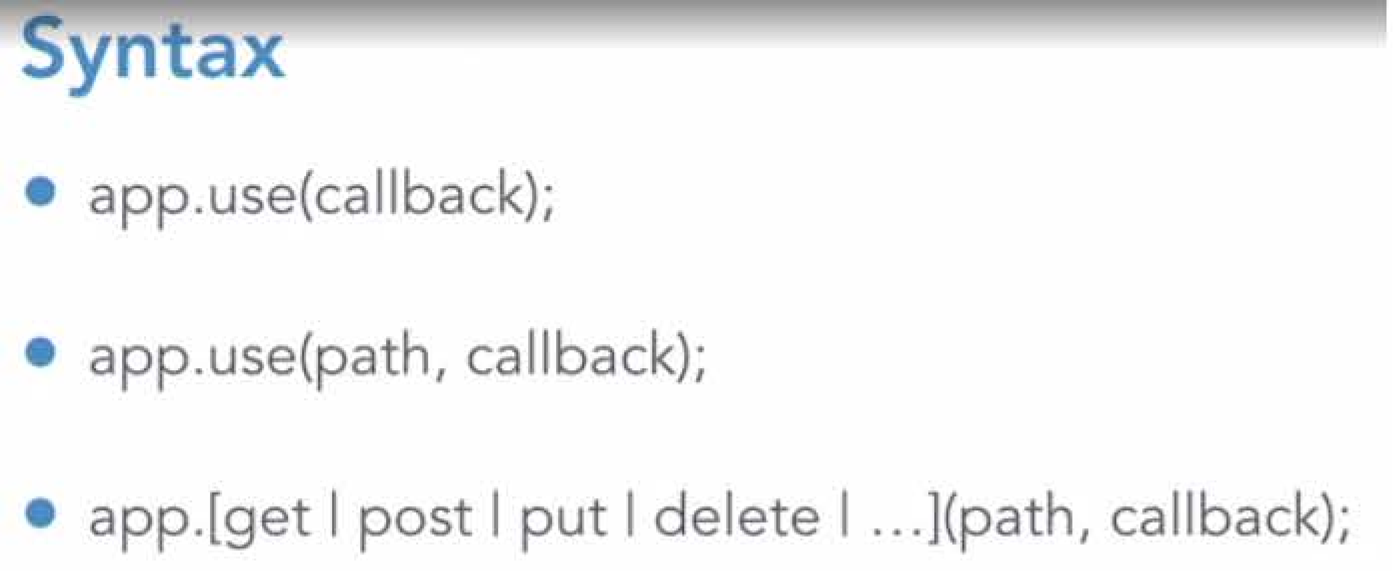
```
<ul>
  <% users.forEach(function(user){ %>
    <%- include('user/show', {user: user}); %>
  <% }); %>
</ul>
```

Custom delimiters

Custom delimiters can be applied on a per-template basis, or globally:

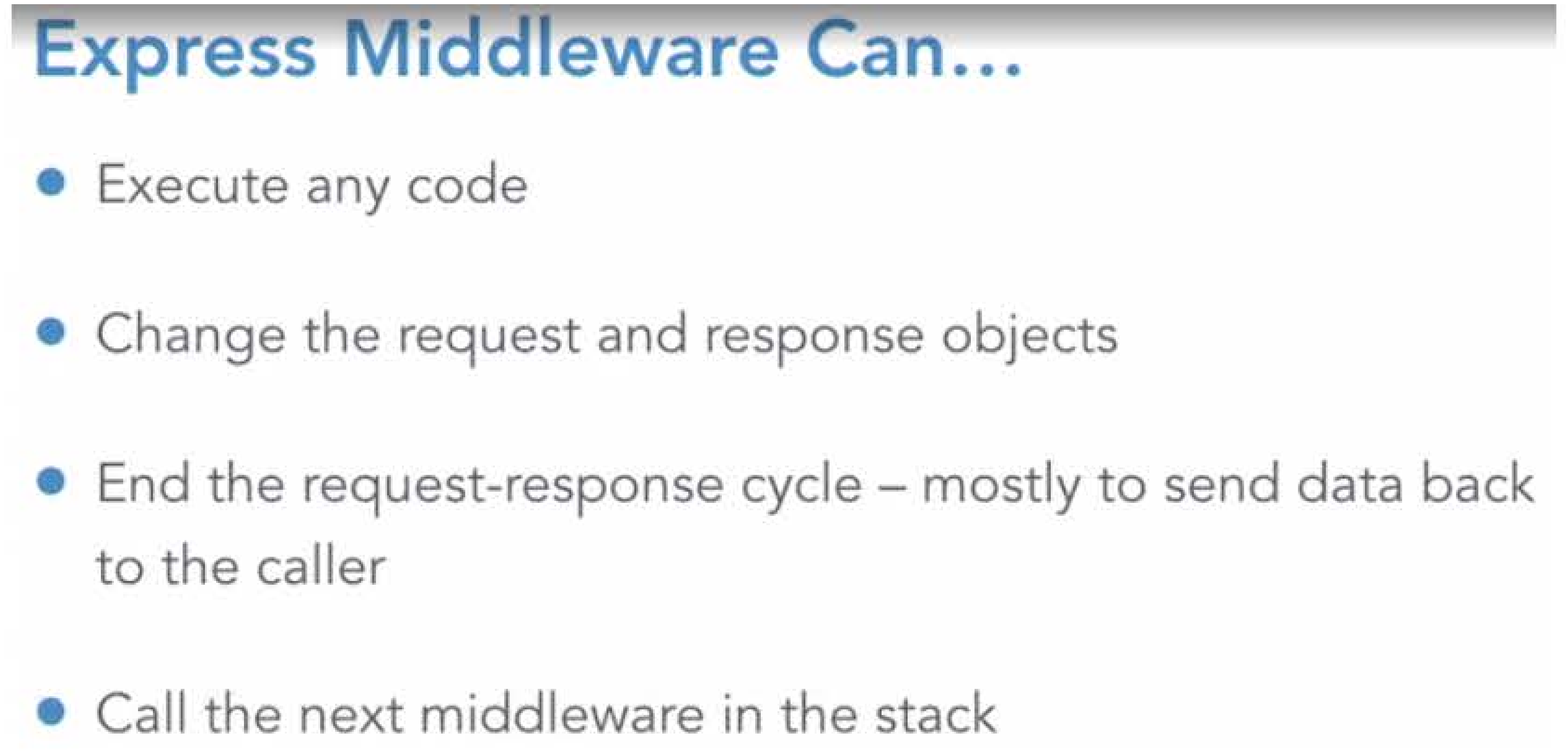
```
let ejs = require('ejs'),
    users = ['geddy', 'neil', 'alex'];

// Just one template
ejs.render('<?= users.join(" | "); ?>', {users: users},
    {delimiter: '?'});
```



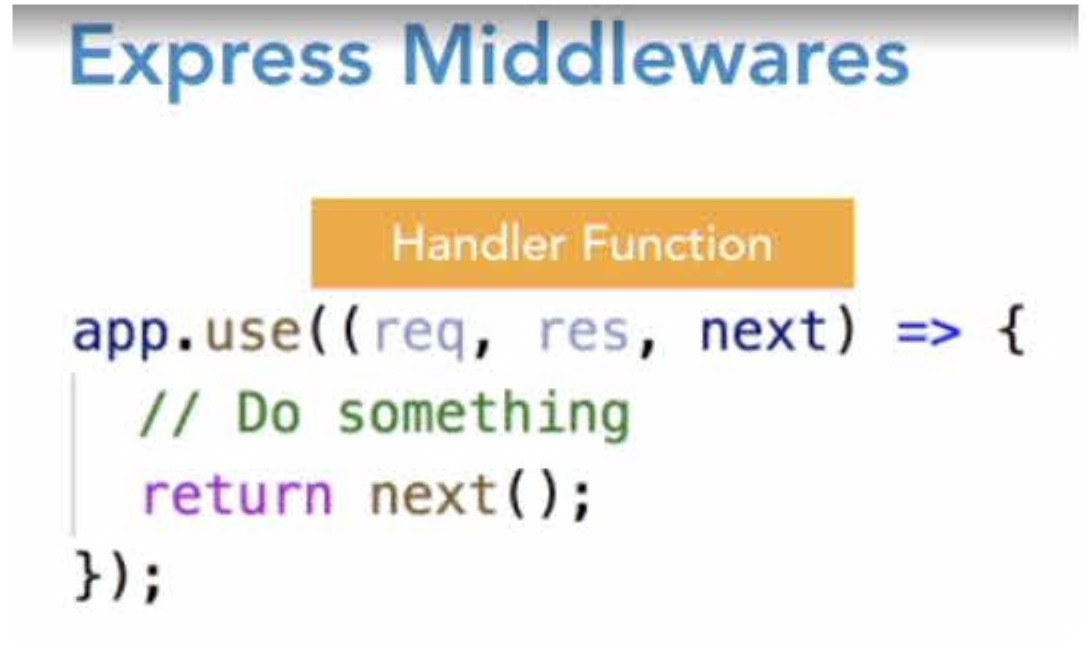
Syntax

- app.use(callback);
- app.use(path, callback);
- app.[get | post | put | delete | …](path, callback);



Express Middleware Can…

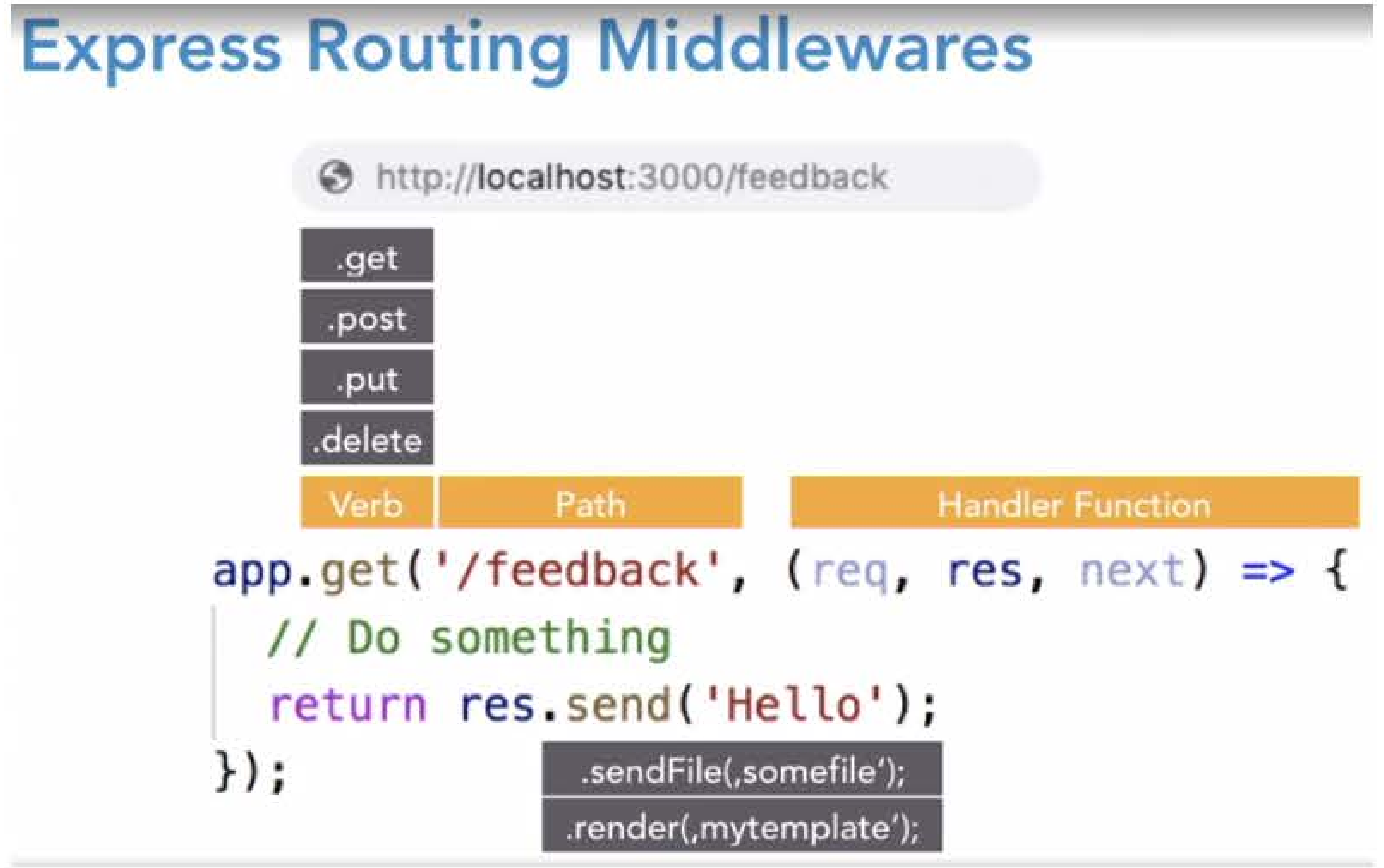
- Execute any code
- Change the request and response objects
- End the request-response cycle – mostly to send data back to the caller
- Call the next middleware in the stack



Express Middlewares

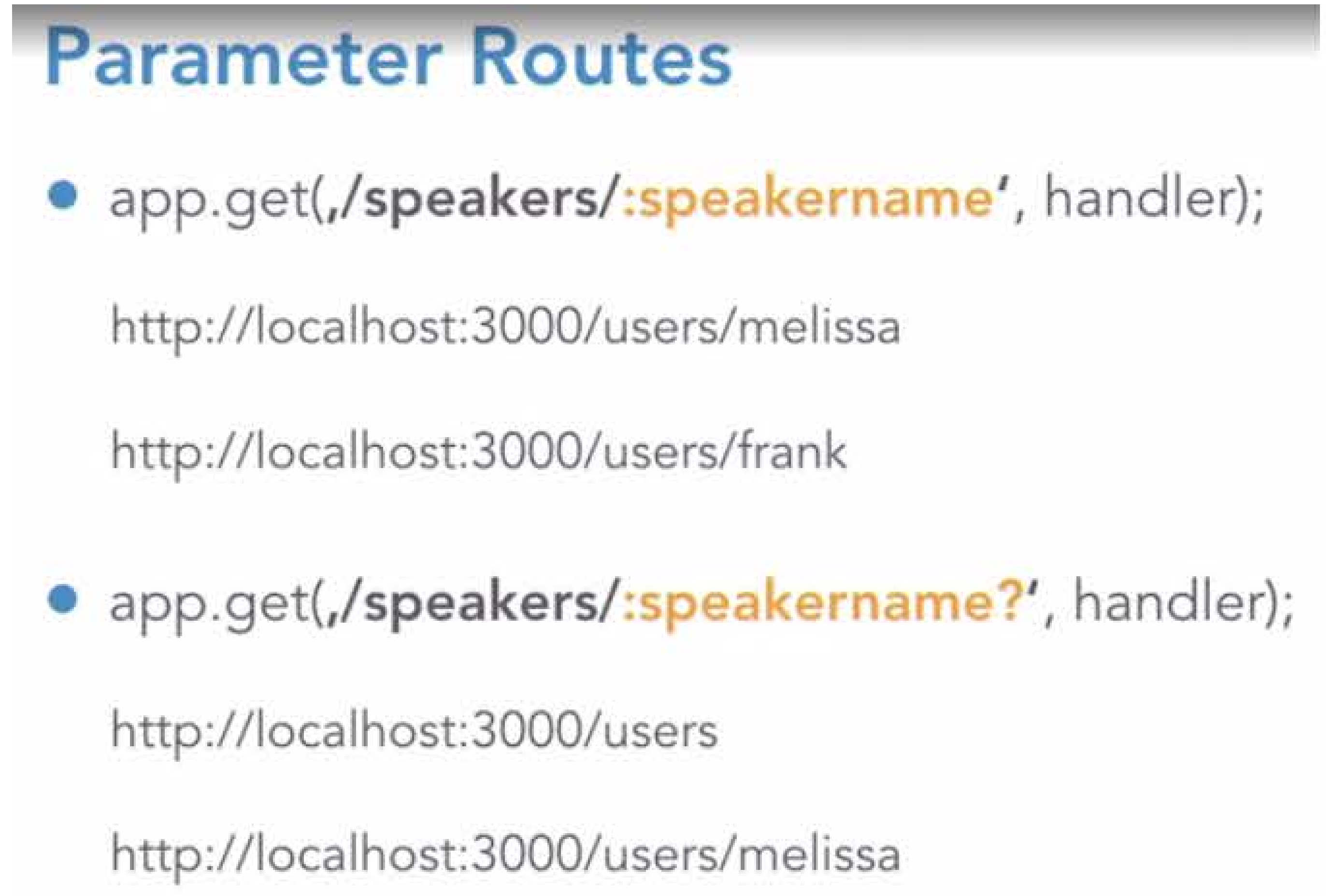
Handler Function

```
app.use((req, res, next) => {
  // Do something
  return next();
});
```



Express Routing Middlewares

```
app.get('/feedback', (req, res, next) => {
  // Do something
  return res.send('Hello');
});
```



Parameter Routes

- app.get(‚**/speakers/:speakername**‘, handler);

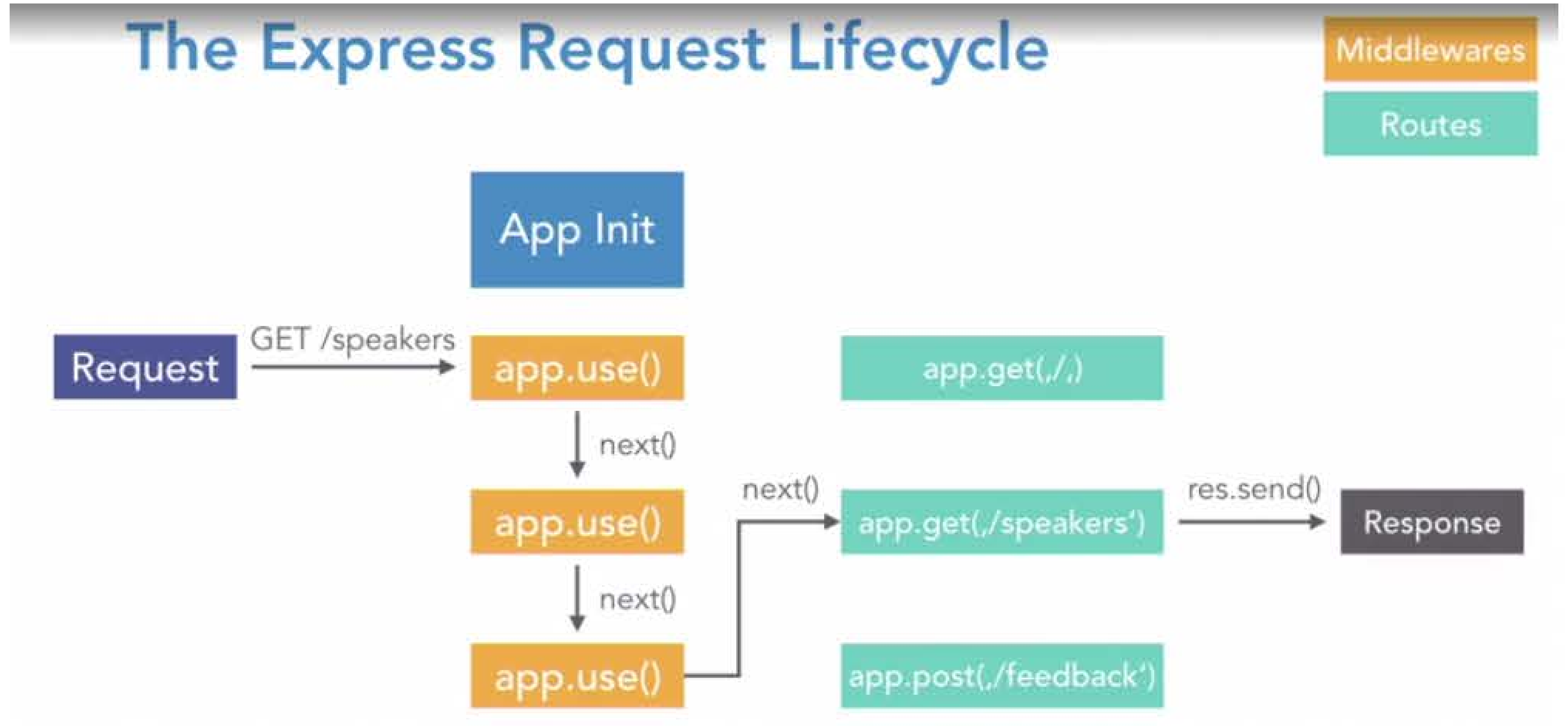
  http://localhost:3000/users/melissa

  http://localhost:3000/users/frank

- app.get(‚**/speakers/:speakername?**‘, handler);

  http://localhost:3000/users

  http://localhost:3000/users/melissa

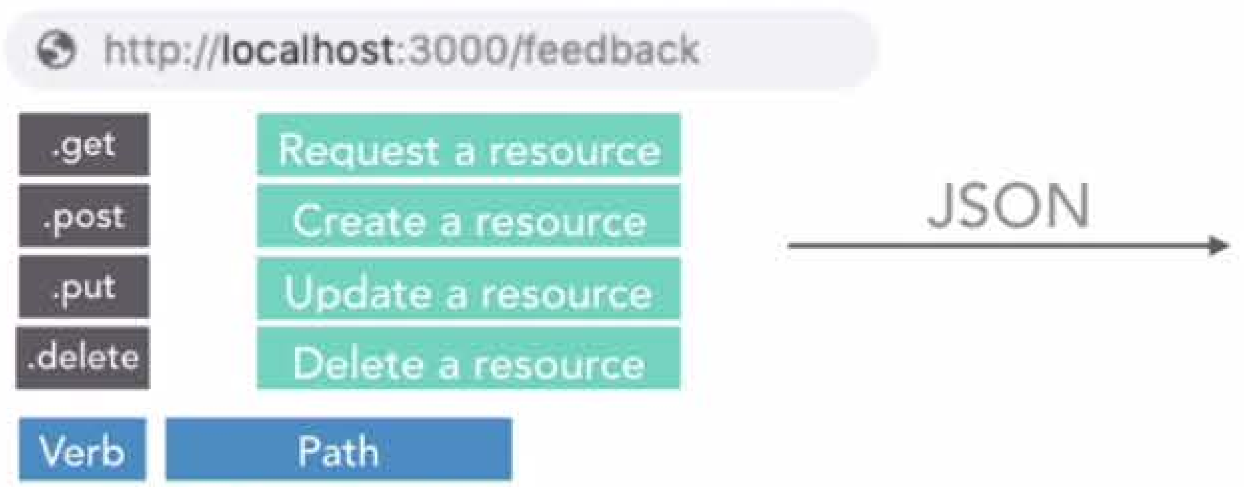


The Express Request Lifecycle

A Little Intro to REST

Representational State Transfer

HTTP Verbs and REST



```
app.get('/feedback', (req, res, next) => {
  // Do something
  return res.send('Hello');
});
```

Create, Update, and Delete (CRUD) via REST

- HTTP **GET** http://mysite.com/users

  Getting all users

- HTTP **POST (payload)** http://mysite.com/users

  Creating a new user

- HTTP **GET** http://mysite.com/users/42

  Getting the user with the ID 42

Create, Update, and Delete (CRUD) via REST

- HTTP **PUT (payload)** http://mysite.com/users/42

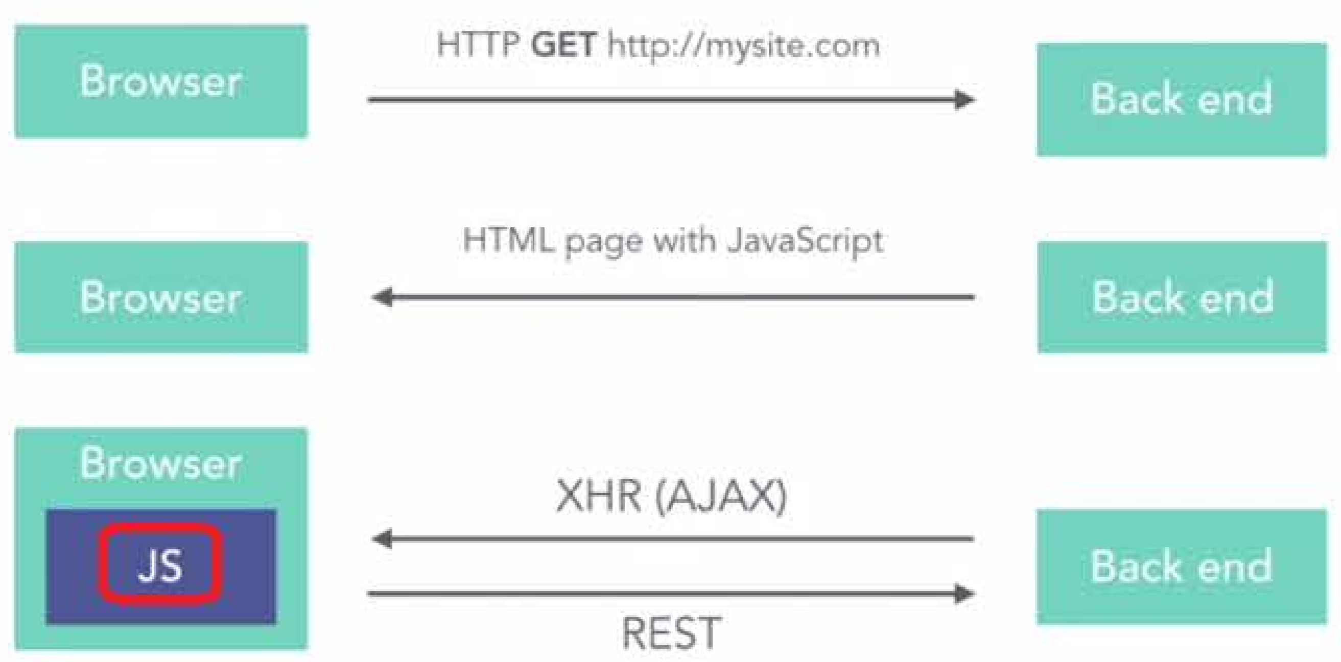
  Getting all users

- HTTP **DELETE** http://mysite.com/users/42

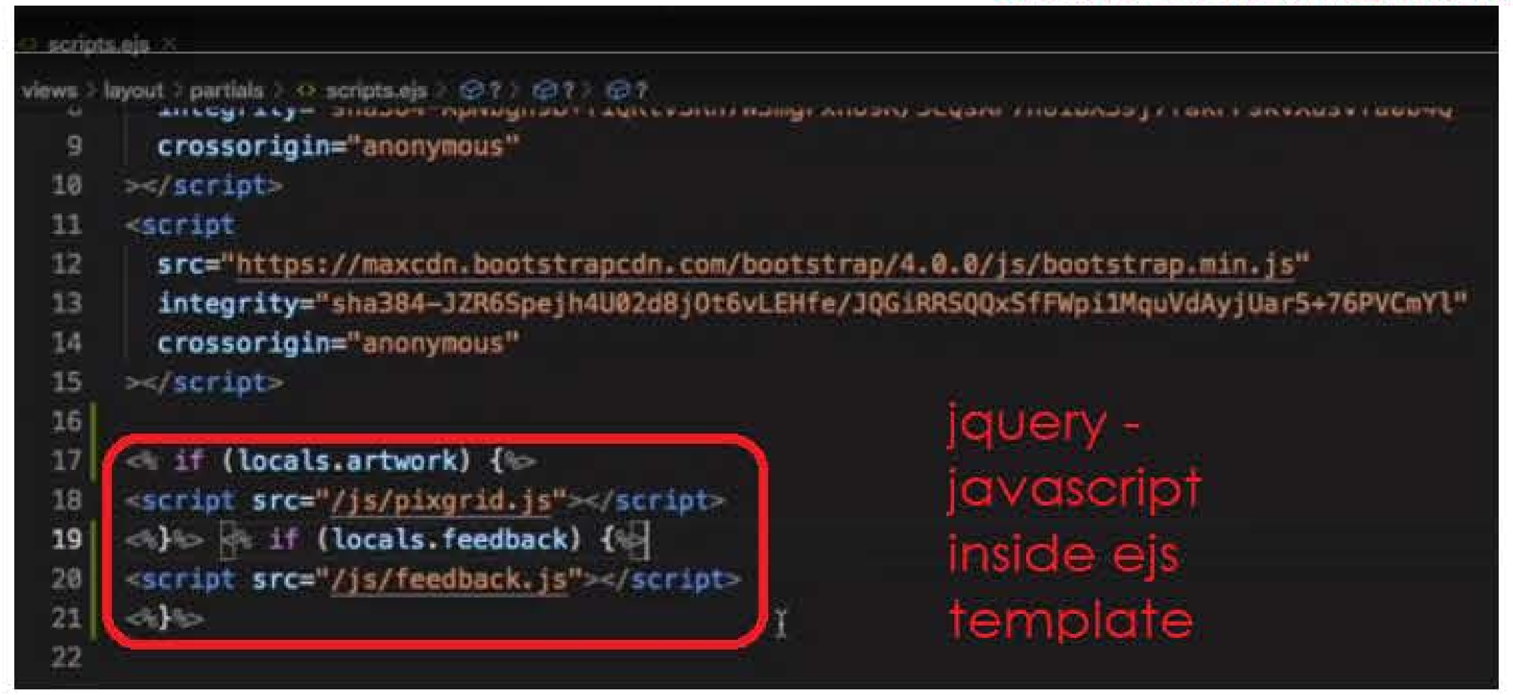
  Deleting the user with the ID 42

REST in Web Applications

or SPA



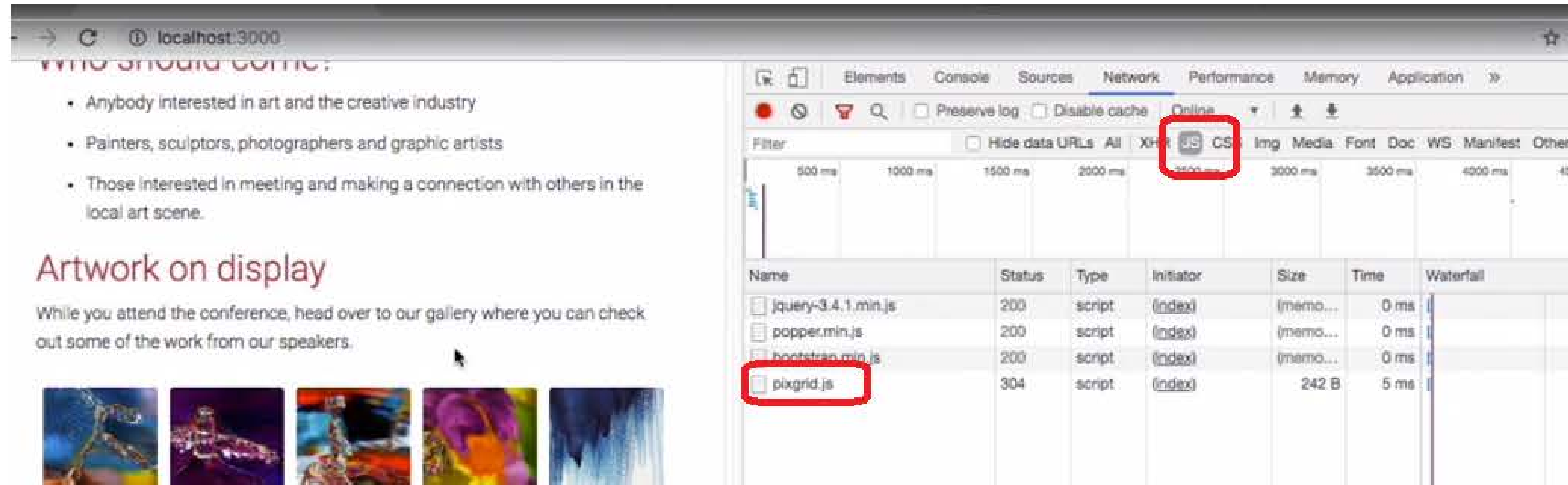
Client side javascript: other client side frameworks can be used such as angular, react or vue.js
jquery is good and easiest solution for few dynamic requests, component is already available thorugh bootstrap



```
  crossorigin="anonymous"
></script>
<script
  src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
  integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
  crossorigin="anonymous"
></script>

<% if (locals.artwork) {%>
<script src="/js/pixgrid.js"></script>
<%}%> <% if (locals.feedback) {%>
<script src="/js/feedback.js"></script>
<%}%>
```

jquery - javascript inside ejs template



- Anybody interested in art and the creative industry
- Painters, sculptors, photographers and graphic artists
- Those interested in meeting and making a connection with others in the local art scene.

Artwork on display

While you attend the conference, head over to our gallery where you can check out some of the work from our speakers.

Name	Status	Type	Initiator	Size	Time	Waterfall
jquery-3.4.1.min.js	200	script	(index)	(memo...	0 ms	
popper.min.js	200	script	(index)	(memo...	0 ms	
bootstrap.min.js	200	script	(index)	(memo...	0 ms	
pixgrid.js	304	script	(index)	242 B	5 ms	