PWA JUMPSTART

@SVENRUPPERT

vaadin}>

PWA JUMPSTART

@SVENRUPPERT

CODING JAVA SINCE 1996

DEVELOPER ADVOCATE

VAADIN

vaadin}>

DESKTOP V. MOBILE

NATIVE DESKTOP APP WINDOWS / OSX / LINUX/..

WEB FOR DESKTOP CHROME / FIREFOX / OPERA / IE / ..

WEB FOR MOBILE ANDROID / IOS / WINDOWS MOBILE / ..

NATIVE MOBILE APP ANDROID / IOS / WINDOWS MOBILE / ..

vaadin}>

WHAT MEANS PWA?

PWA

$\downarrow$

PROGRESSIVE WEB APP

PROGRESSIVE WEB APP

HTML5

CSS3

JAVASCRIPT

PROGRESSIVE WEB APP

SERVICEWORKER

OFFLINE-FUNCTIONS

CACHING

PUSH

vaadin}>

PROGRESSIVE WEB APP

SERVICEWORKER

Optional

OFFLINE-FUNCTIONS

CACHING

PUSH

vaadin}>

PROGRESSIVE WEB APP

WEBSERVER HTTPS WEBCLIENT

Even during development time

PROGRESSIVE WEB APP

ADD TO HOMESCREEN

ICON ON HOMESCREEN

LOOKS LIKE AN APP

vaadin}>

PROGRESSIVE WEB APP

CONNECTIVITY INDEPENDENT

PROGRESSIVE

RESPONSIVE

APP-LIKE

SAFE

FRESH

RE-ENGAGEABLE

DISCOVERABLE

LIN KABLE

INSTALLABLE

WHAT DO I NEED ?

WHAT DO I NEED ?

A BROWSER THAT WILL SUPPORT IT

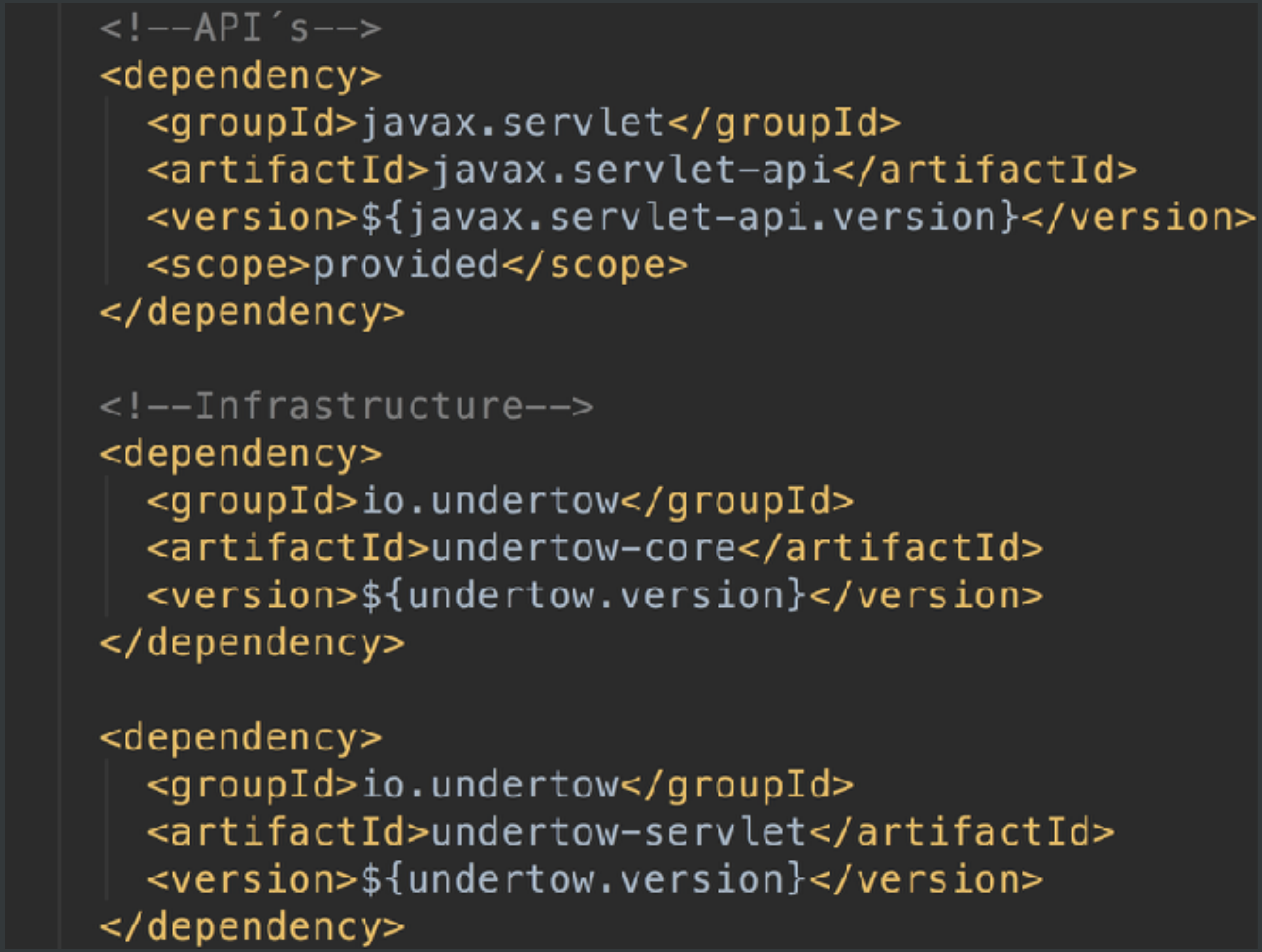
CHROME	✓	EDGE	✗
FIREFOX	✓	SAFARI	✗
OPERA	✓		

vaadin}>

CORE VAADIN APP

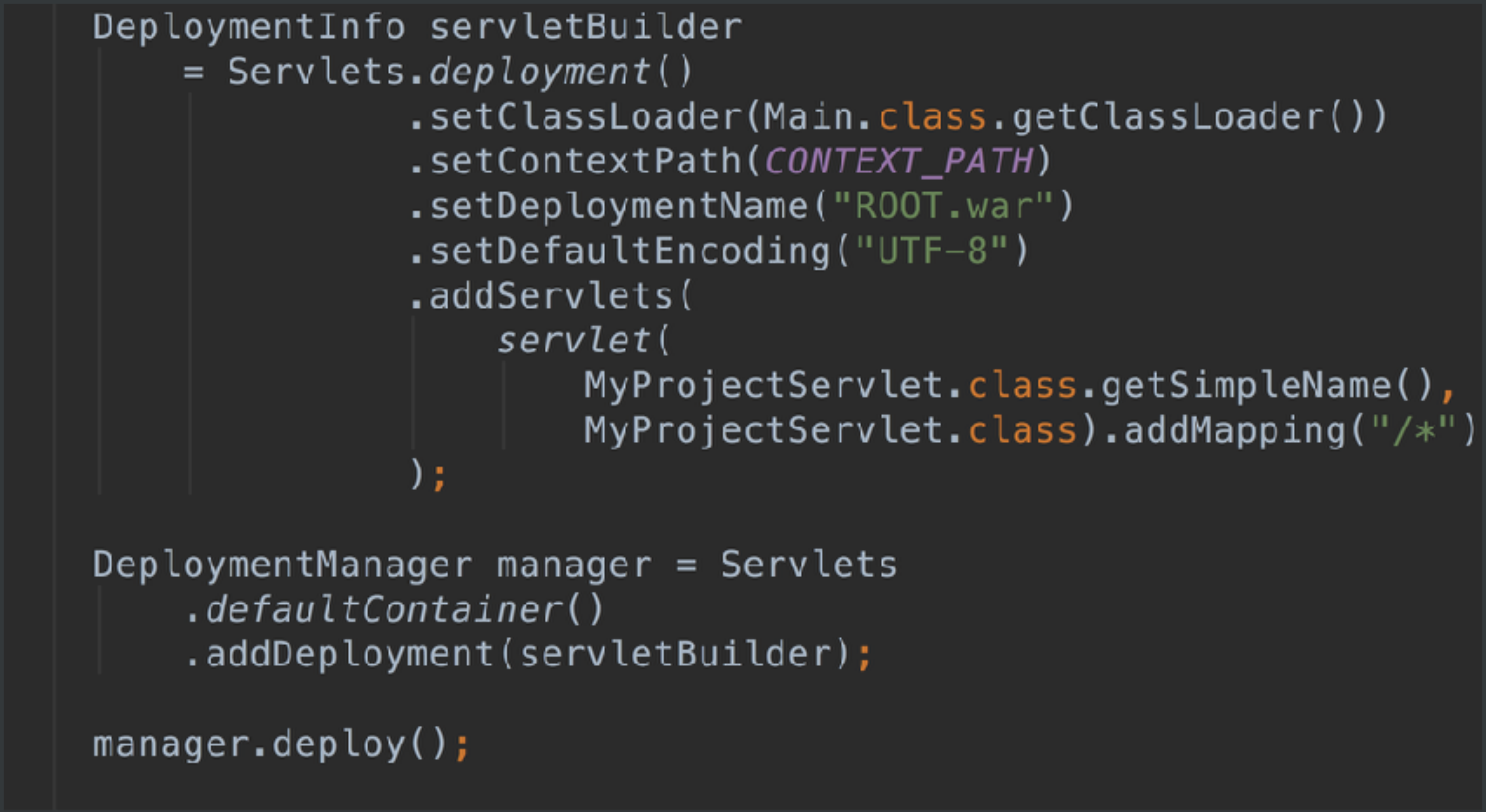
SERVLETCONTAINER

BASED ON UNDERTOW



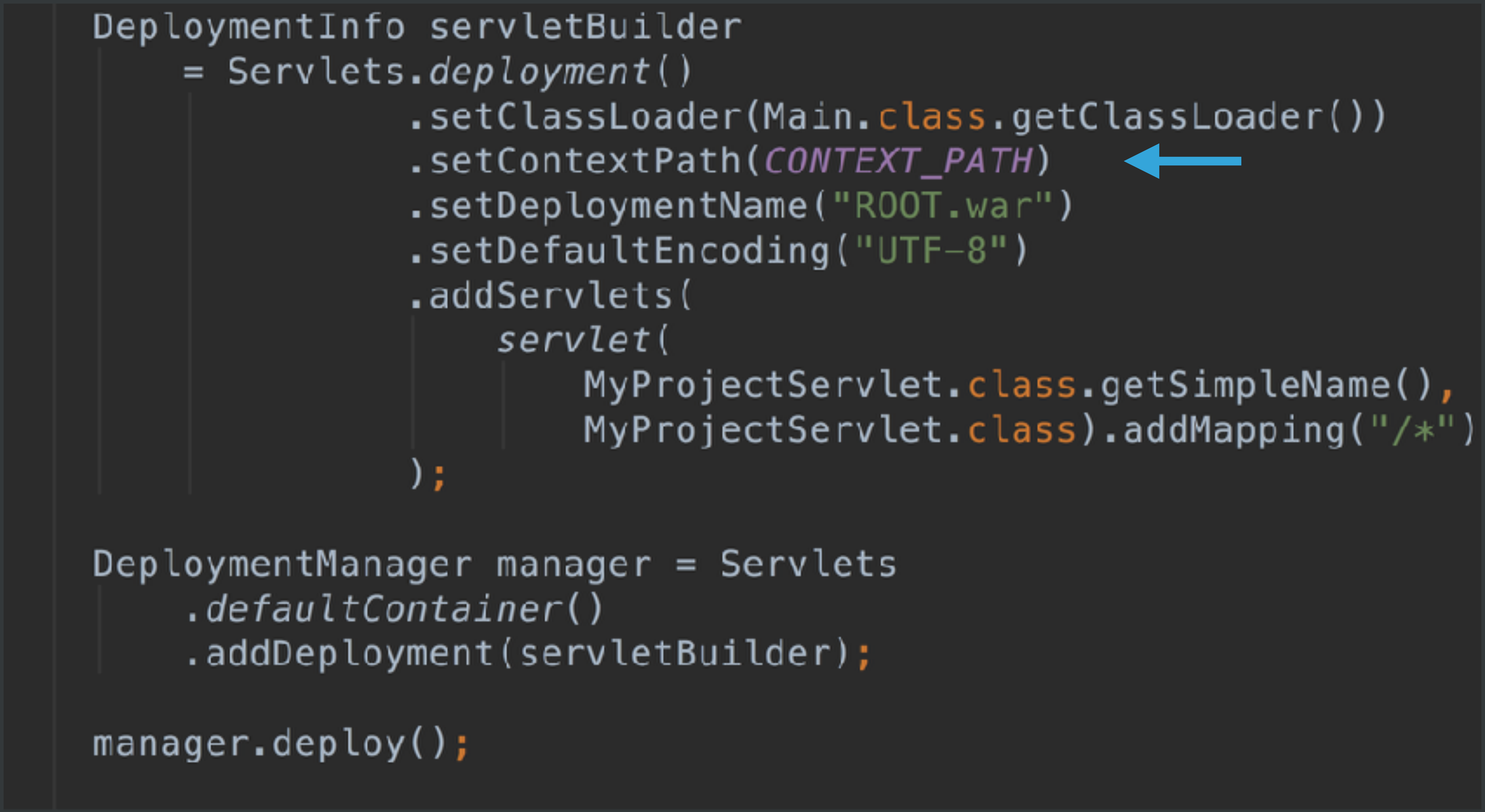
```xml
<!--API´s-->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>${javax.servlet-api.version}</version>
  <scope>provided</scope>
</dependency>

<!--Infrastructure-->
<dependency>
  <groupId>io.undertow</groupId>
  <artifactId>undertow-core</artifactId>
  <version>${undertow.version}</version>
</dependency>

<dependency>
  <groupId>io.undertow</groupId>
  <artifactId>undertow-servlet</artifactId>
  <version>${undertow.version}</version>
</dependency>
```

BASED ON UNDERTOW



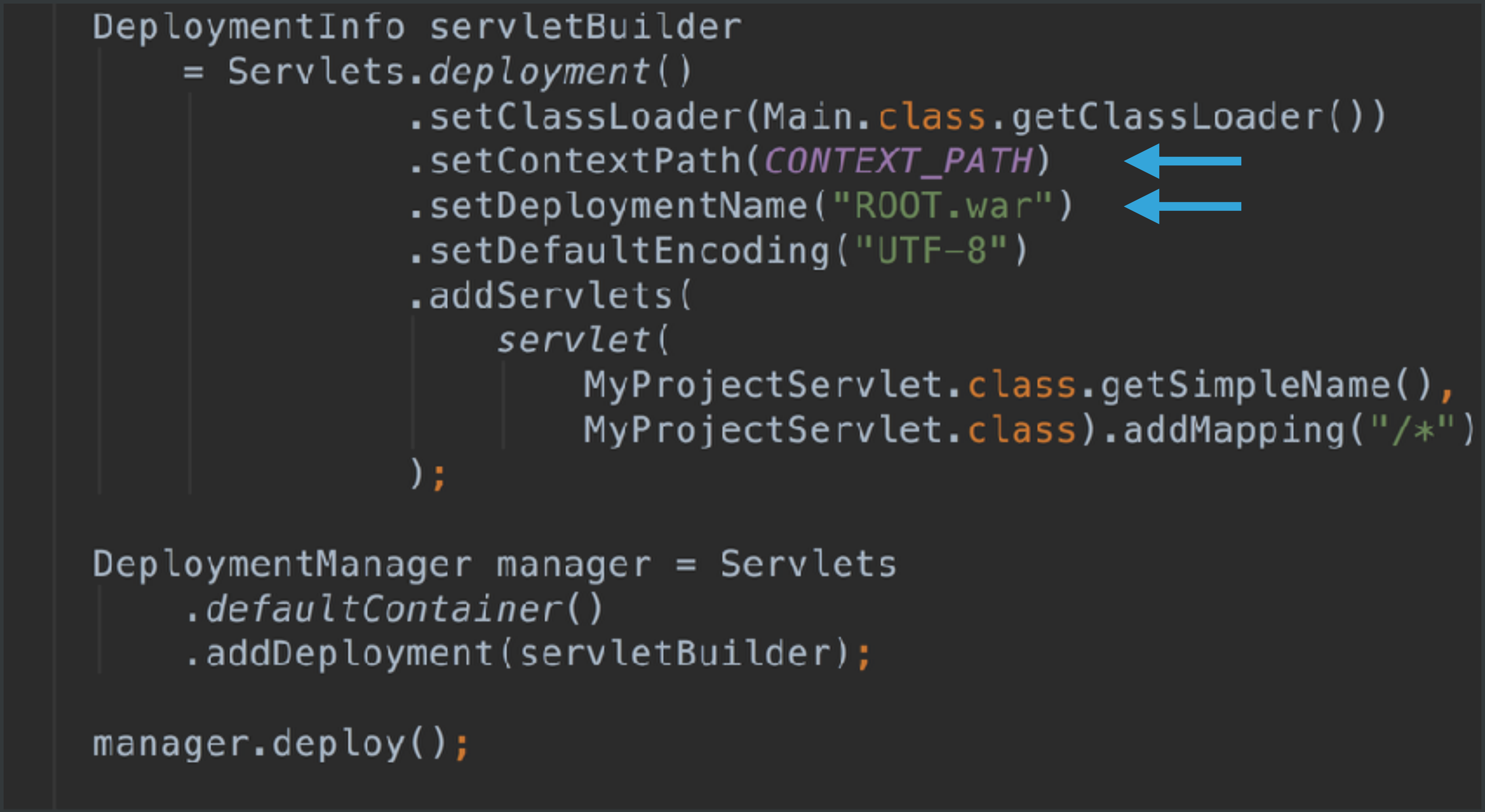
```java
DeploymentInfo servletBuilder
    = Servlets.deployment()
              .setClassLoader(Main.class.getClassLoader())
              .setContextPath(CONTEXT_PATH)
              .setDeploymentName("ROOT.war")
              .setDefaultEncoding("UTF-8")
              .addServlets(
                  servlet(
                      MyProjectServlet.class.getSimpleName(),
                      MyProjectServlet.class).addMapping("/*")
              );

DeploymentManager manager = Servlets
    .defaultContainer()
    .addDeployment(servletBuilder);

manager.deploy();
```

BASED ON UNDERTOW



```
DeploymentInfo servletBuilder
    = Servlets.deployment()
              .setClassLoader(Main.class.getClassLoader())
              .setContextPath(CONTEXT_PATH)
              .setDeploymentName("ROOT.war")
              .setDefaultEncoding("UTF-8")
              .addServlets(
                  servlet(
                      MyProjectServlet.class.getSimpleName(),
                      MyProjectServlet.class).addMapping("/*")
              );

DeploymentManager manager = Servlets
    .defaultContainer()
    .addDeployment(servletBuilder);

manager.deploy();
```

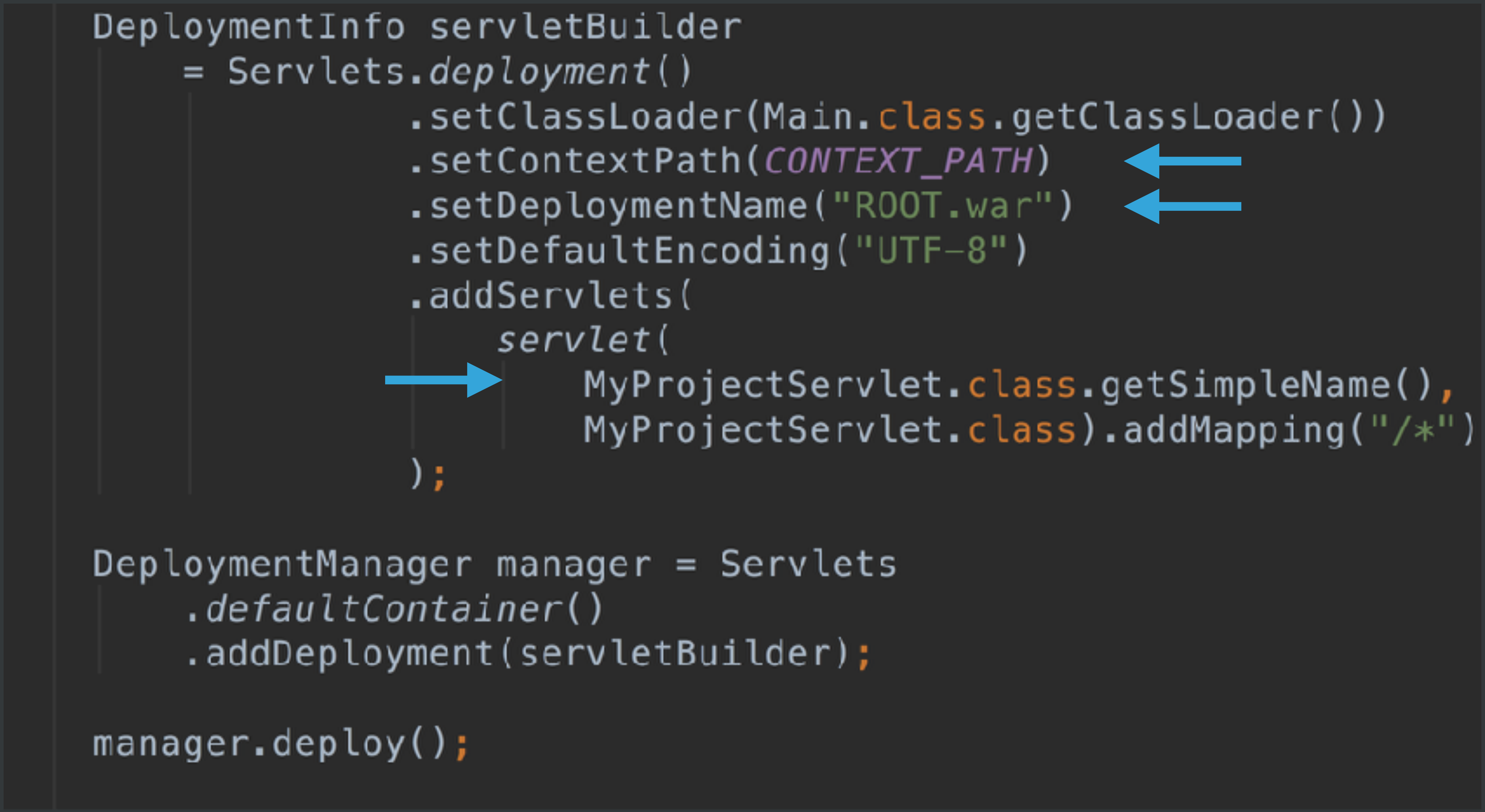
BASED ON UNDERTOW



```
DeploymentInfo servletBuilder
    = Servlets.deployment()
            .setClassLoader(Main.class.getClassLoader())
            .setContextPath(CONTEXT_PATH)
            .setDeploymentName("ROOT.war")
            .setDefaultEncoding("UTF-8")
            .addServlets(
                servlet(
                    MyProjectServlet.class.getSimpleName(),
                    MyProjectServlet.class).addMapping("/*")
            );

DeploymentManager manager = Servlets
    .defaultContainer()
    .addDeployment(servletBuilder);

manager.deploy();
```

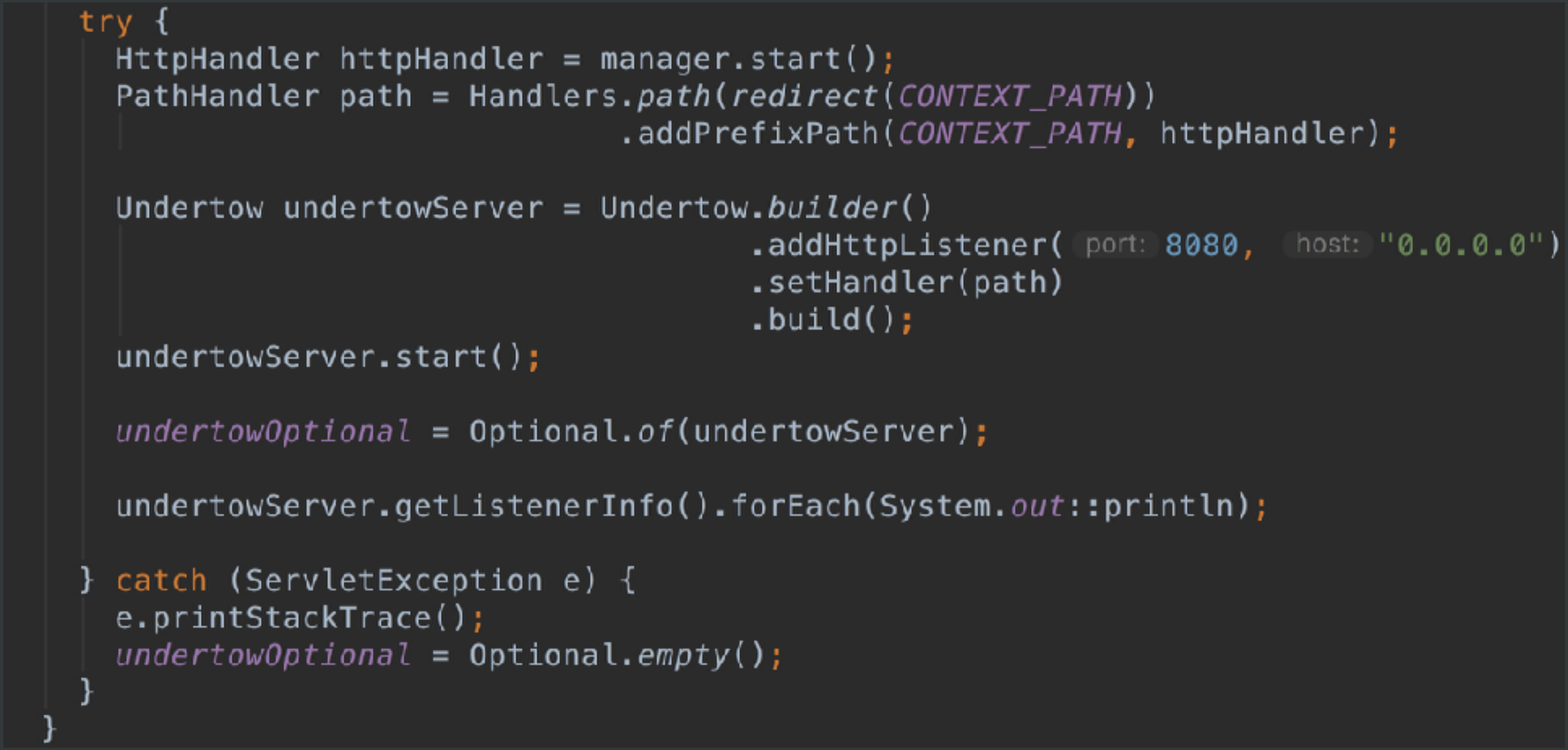
vaadin}>

BASED ON UNDERTOW



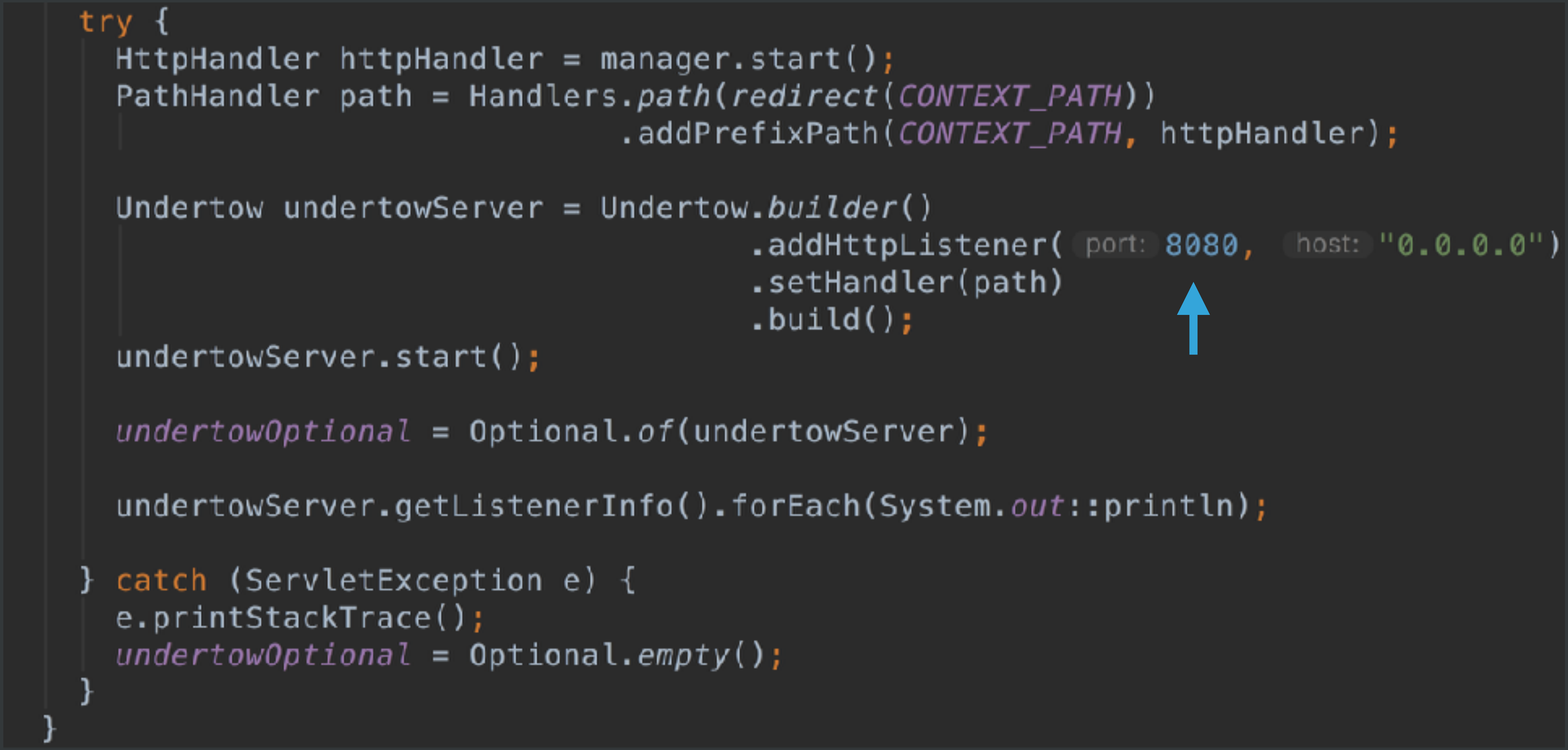
```
DeploymentInfo servletBuilder
    = Servlets.deployment()
                .setClassLoader(Main.class.getClassLoader())
                .setContextPath(CONTEXT_PATH)
                .setDeploymentName("ROOT.war")
                .setDefaultEncoding("UTF-8")
                .addServlets(
                    servlet(
                        MyProjectServlet.class.getSimpleName(),
                        MyProjectServlet.class).addMapping("/*")
                );

DeploymentManager manager = Servlets
    .defaultContainer()
    .addDeployment(servletBuilder);

manager.deploy();
```

BASED ON UNDERTOW



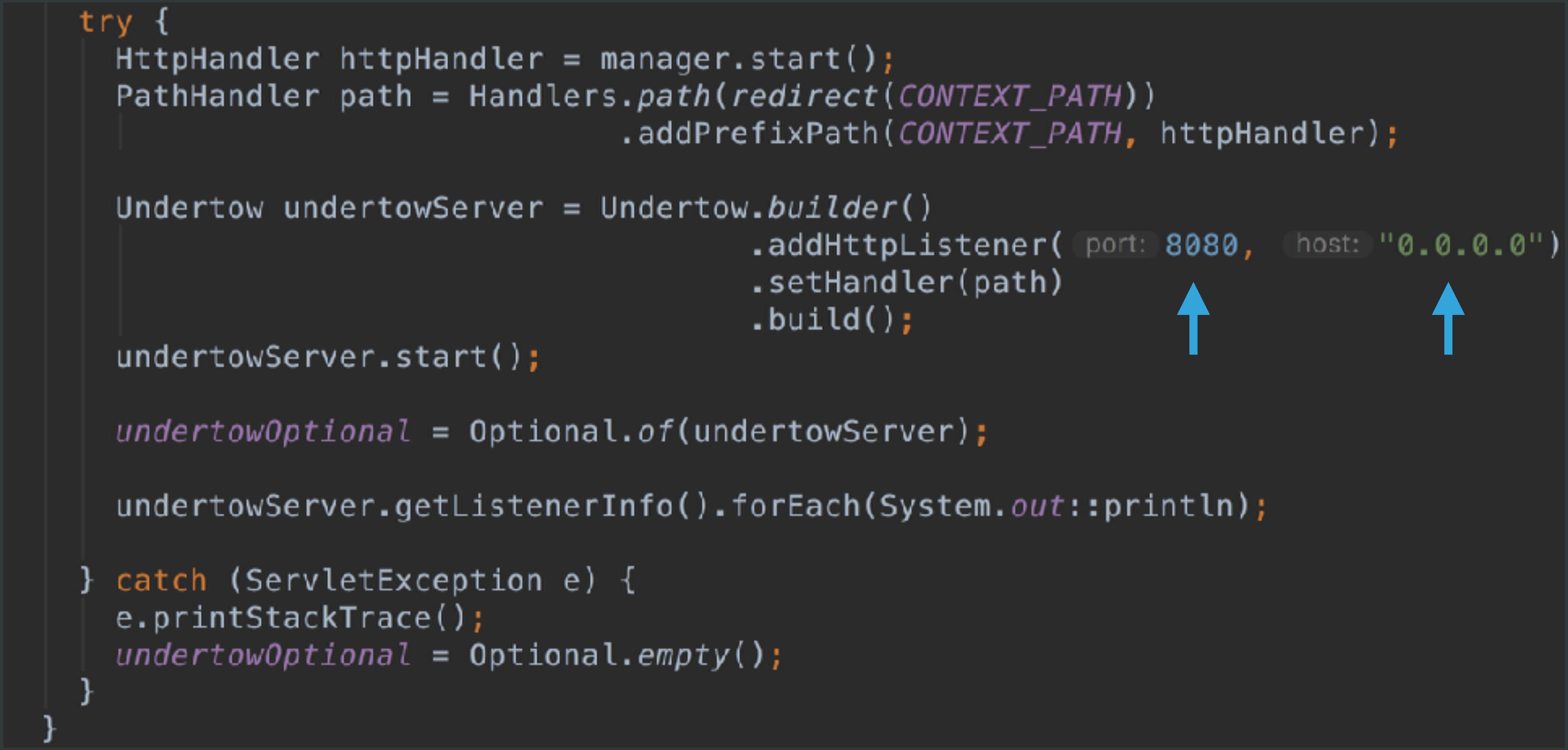
```
  try {
    HttpHandler httpHandler = manager.start();
    PathHandler path = Handlers.path(redirect(CONTEXT_PATH))
                           .addPrefixPath(CONTEXT_PATH, httpHandler);

    Undertow undertowServer = Undertow.builder()
                                  .addHttpListener( port: 8080,  host: "0.0.0.0")
                                  .setHandler(path)
                                  .build();
    undertowServer.start();

    undertowOptional = Optional.of(undertowServer);

    undertowServer.getListenerInfo().forEach(System.out::println);

  } catch (ServletException e) {
    e.printStackTrace();
    undertowOptional = Optional.empty();
  }
}
```

BASED ON UNDERTOW



```
    try {
        HttpHandler httpHandler = manager.start();
        PathHandler path = Handlers.path(redirect(CONTEXT_PATH))
                                   .addPrefixPath(CONTEXT_PATH, httpHandler);

        Undertow undertowServer = Undertow.builder()
                                          .addHttpListener( port: 8080,  host: "0.0.0.0")
                                          .setHandler(path)
                                          .build();
        undertowServer.start();

        undertowOptional = Optional.of(undertowServer);

        undertowServer.getListenerInfo().forEach(System.out::println);

    } catch (ServletException e) {
        e.printStackTrace();
        undertowOptional = Optional.empty();
    }
}
```

BASED ON UNDERTOW

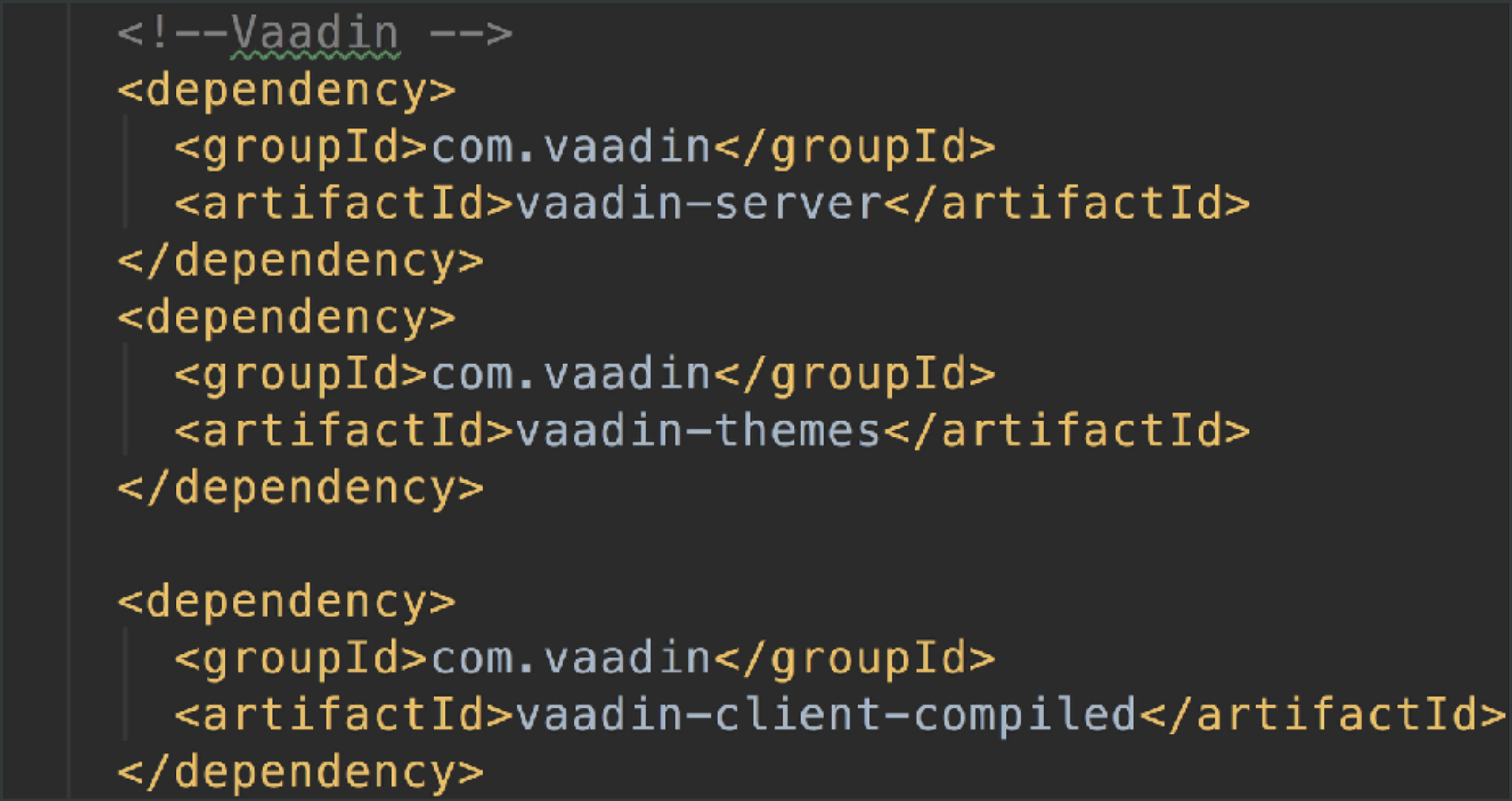


```
    try {
        HttpHandler httpHandler = manager.start();
        PathHandler path = Handlers.path(redirect(CONTEXT_PATH))
                                   .addPrefixPath(CONTEXT_PATH, httpHandler);

        Undertow undertowServer = Undertow.builder()
                                          .addHttpListener( port: 8080,  host: "0.0.0.0")
                                          .setHandler(path)
                                          .build();
        undertowServer.start();

        undertowOptional = Optional.of(undertowServer);

        undertowServer.getListenerInfo().forEach(System.out::println);

    } catch (ServletException e) {
        e.printStackTrace();
        undertowOptional = Optional.empty();
    }
}
```

vaadin}>

V A A D I N

SERVLET

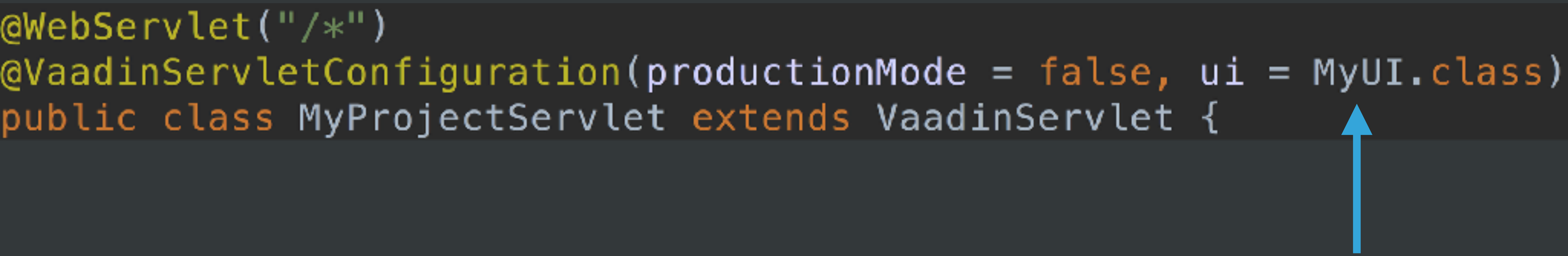


```xml
<!--Vaadin -->
<dependency>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-server</artifactId>
</dependency>
<dependency>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-themes</artifactId>
</dependency>

<dependency>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-client-compiled</artifactId>
</dependency>
```

SERVLET

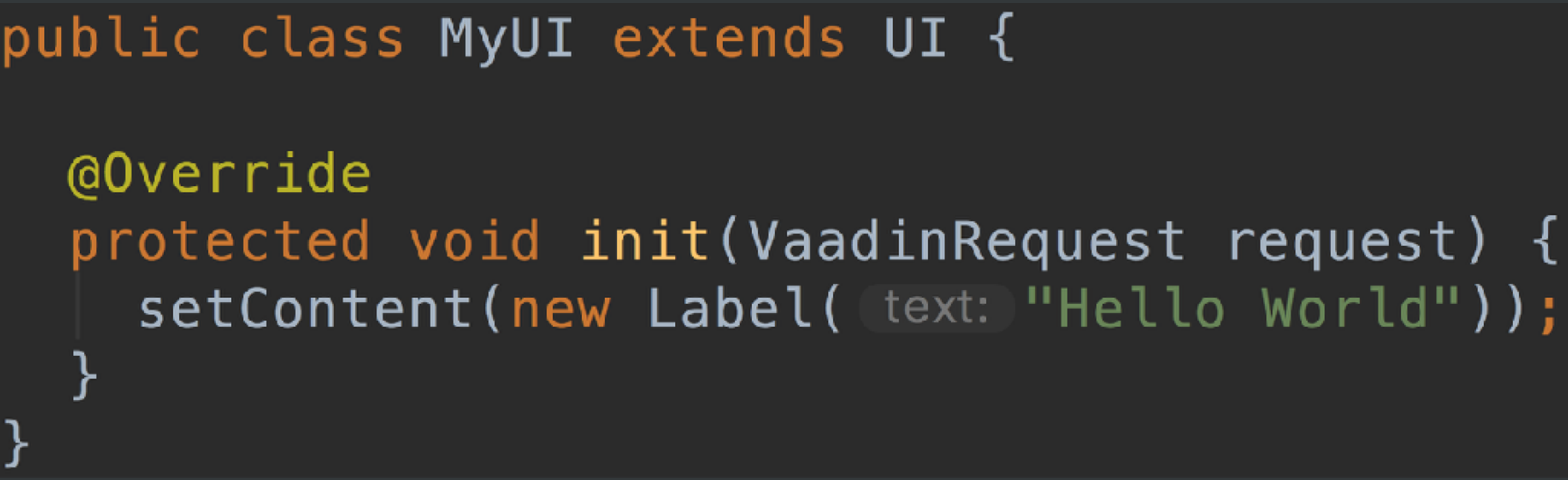
```java
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {
```

SERVLET



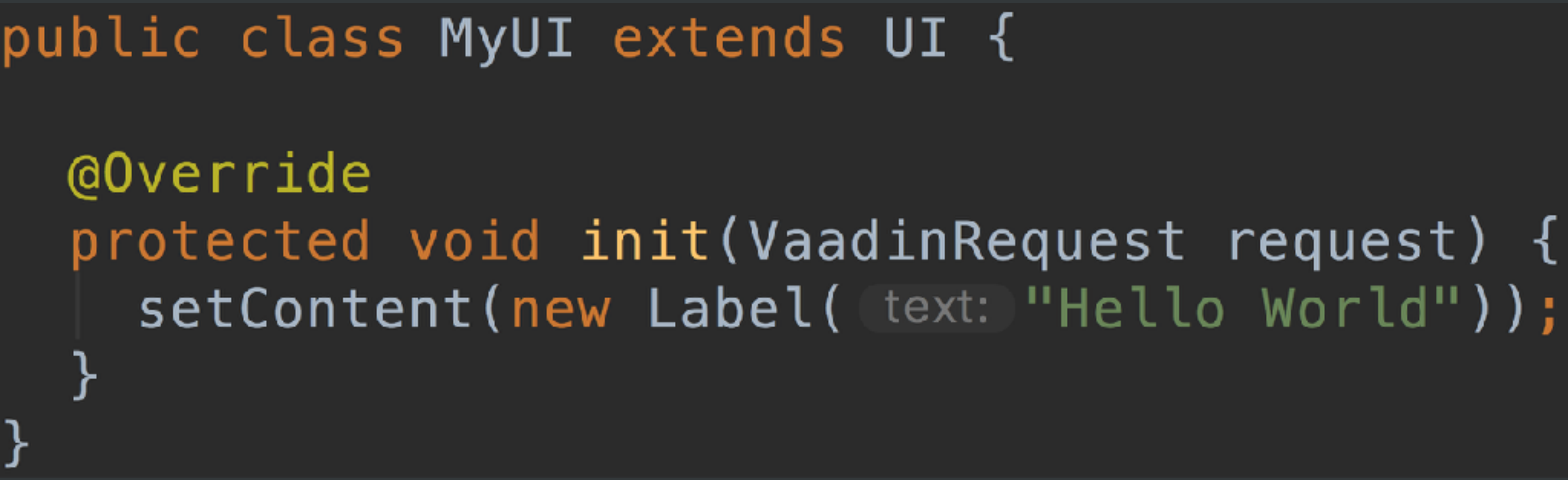
```
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {
```

UI



```
public class MyUI extends UI {

  @Override
  protected void init(VaadinRequest request) {
    setContent(new Label( text: "Hello World"));
  }
}
```

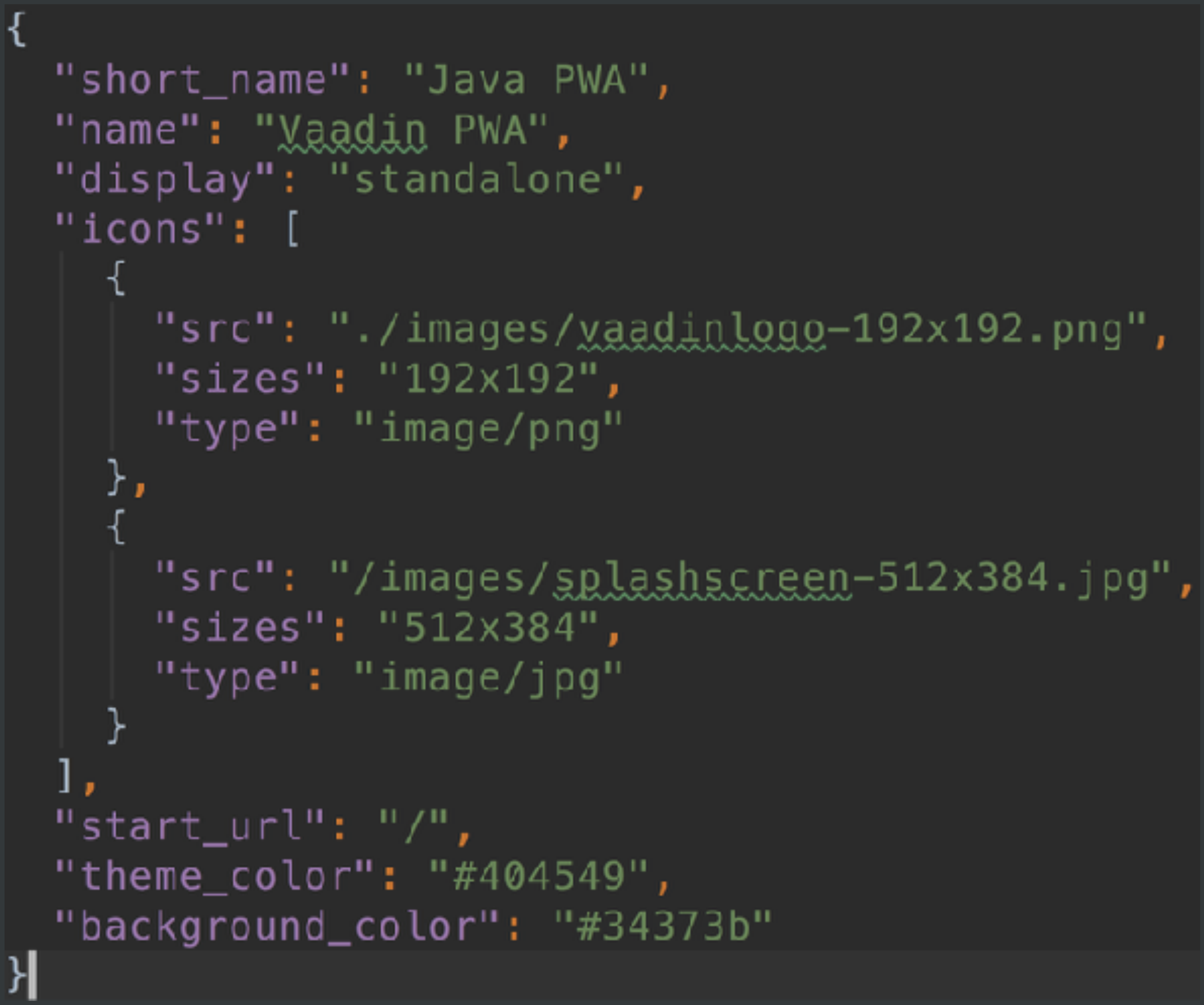
UI



```
public class MyUI extends UI {

  @Override
  protected void init(VaadinRequest request) {
    setContent(new Label( text: "Hello World"));
  }
}
```

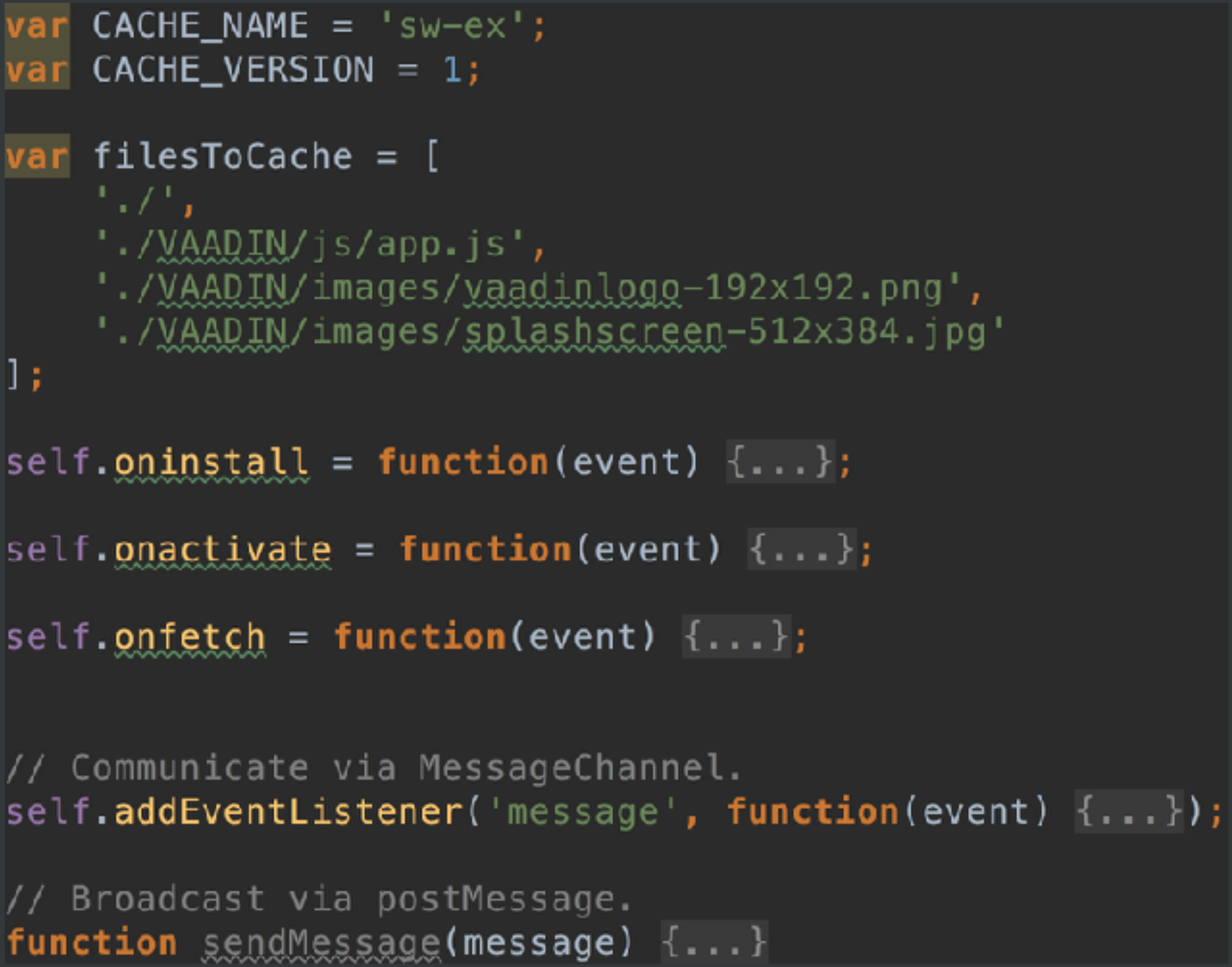
APP -> PWA

MANIFEST.JSON



```json
{
  "short_name": "Java PWA",
  "name": "Vaadin PWA",
  "display": "standalone",
  "icons": [
    {
      "src": "./images/vaadinlogo-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/images/splashscreen-512x384.jpg",
      "sizes": "512x384",
      "type": "image/jpg"
    }
  ],
  "start_url": "/",
  "theme_color": "#404549",
  "background_color": "#34373b"
}
```

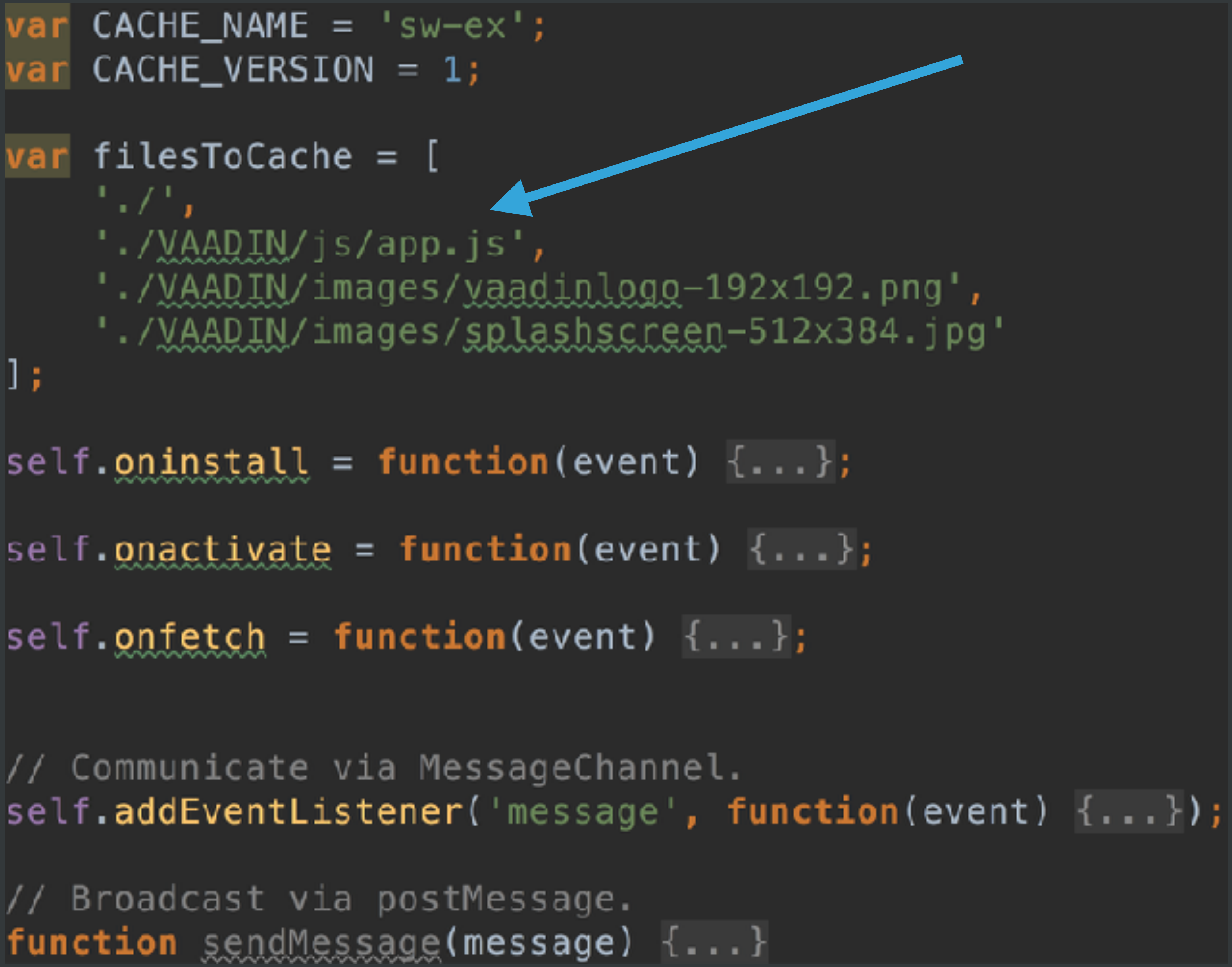
SW.JS



```
var CACHE_NAME = 'sw-ex';
var CACHE_VERSION = 1;

var filesToCache = [
    './',
    './VAADIN/js/app.js',
    './VAADIN/images/vaadinlogo-192x192.png',
    './VAADIN/images/splashscreen-512x384.jpg'
];

self.oninstall = function(event) {...};

self.onactivate = function(event) {...};

self.onfetch = function(event) {...};


// Communicate via MessageChannel.
self.addEventListener('message', function(event) {...});

// Broadcast via postMessage.
function sendMessage(message) {...}
```

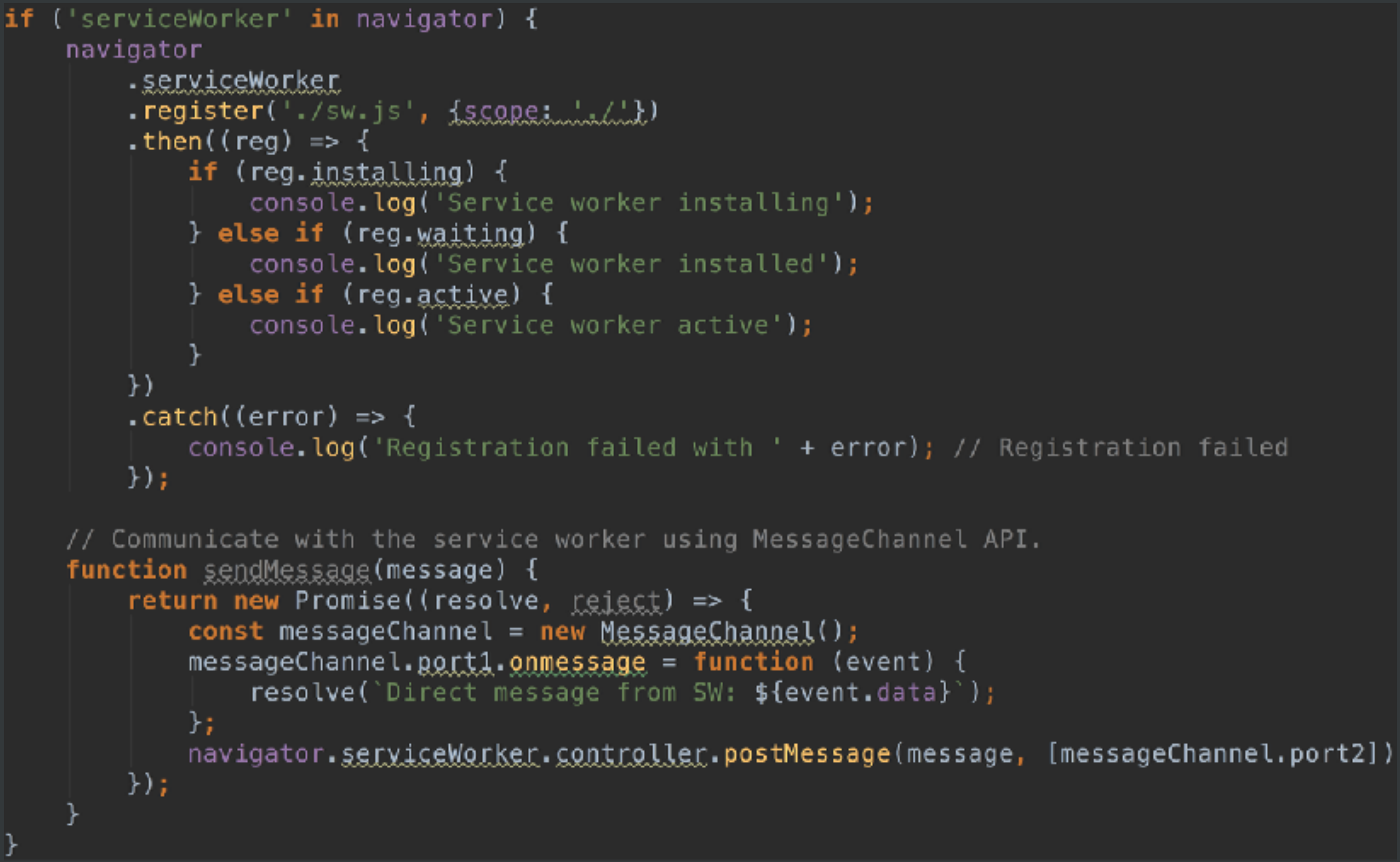
vaadin}>

SW.JS



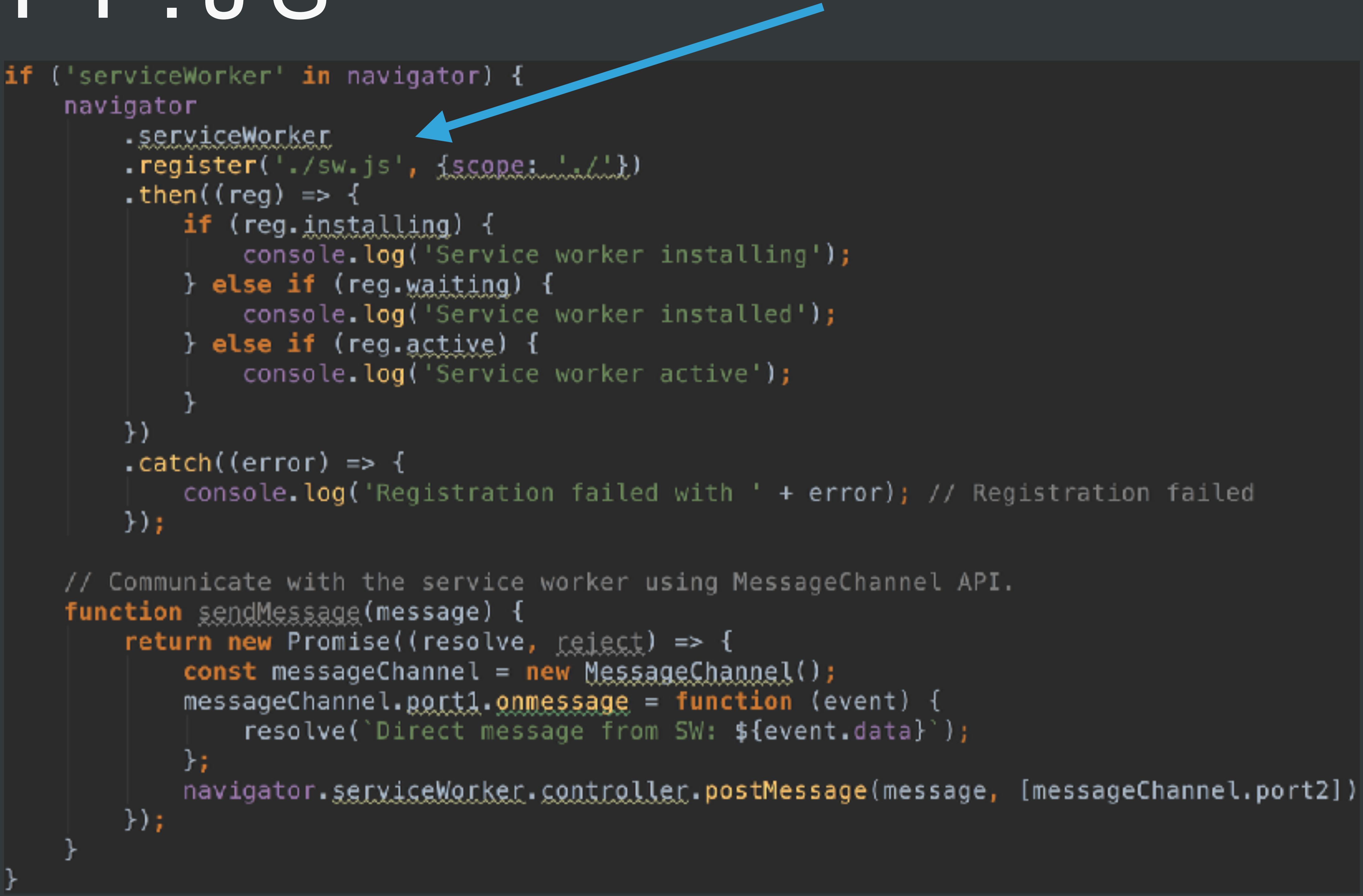
```
var CACHE_NAME = 'sw-ex';
var CACHE_VERSION = 1;

var filesToCache = [
    './',
    './VAADIN/js/app.js',
    './VAADIN/images/vaadinlogo-192x192.png',
    './VAADIN/images/splashscreen-512x384.jpg'
];

self.oninstall = function(event) {...};

self.onactivate = function(event) {...};

self.onfetch = function(event) {...};


// Communicate via MessageChannel.
self.addEventListener('message', function(event) {...});

// Broadcast via postMessage.
function sendMessage(message) {...}
```

APP.JS



```
if ('serviceWorker' in navigator) {
    navigator
        .serviceWorker
        .register('./sw.js', {scope: './'})
        .then((reg) => {
            if (reg.installing) {
                console.log('Service worker installing');
            } else if (reg.waiting) {
                console.log('Service worker installed');
            } else if (reg.active) {
                console.log('Service worker active');
            }
        })
        .catch((error) => {
            console.log('Registration failed with ' + error); // Registration failed
        });

    // Communicate with the service worker using MessageChannel API.
    function sendMessage(message) {
        return new Promise((resolve, reject) => {
            const messageChannel = new MessageChannel();
            messageChannel.port1.onmessage = function (event) {
                resolve(`Direct message from SW: ${event.data}`);
            };
            navigator.serviceWorker.controller.postMessage(message, [messageChannel.port2])
        });
    }
}
```

APP.JS



```
if ('serviceWorker' in navigator) {
    navigator
        .serviceWorker
        .register('./sw.js', {scope: './'})
        .then((reg) => {
            if (reg.installing) {
                console.log('Service worker installing');
            } else if (reg.waiting) {
                console.log('Service worker installed');
            } else if (reg.active) {
                console.log('Service worker active');
            }
        })
        .catch((error) => {
            console.log('Registration failed with ' + error); // Registration failed
        });

    // Communicate with the service worker using MessageChannel API.
    function sendMessage(message) {
        return new Promise((resolve, reject) => {
            const messageChannel = new MessageChannel();
            messageChannel.port1.onmessage = function (event) {
                resolve(`Direct message from SW: ${event.data}`);
            };
            navigator.serviceWorker.controller.postMessage(message, [messageChannel.port2])
        });
    }
}
```

DEVELOP A PWA

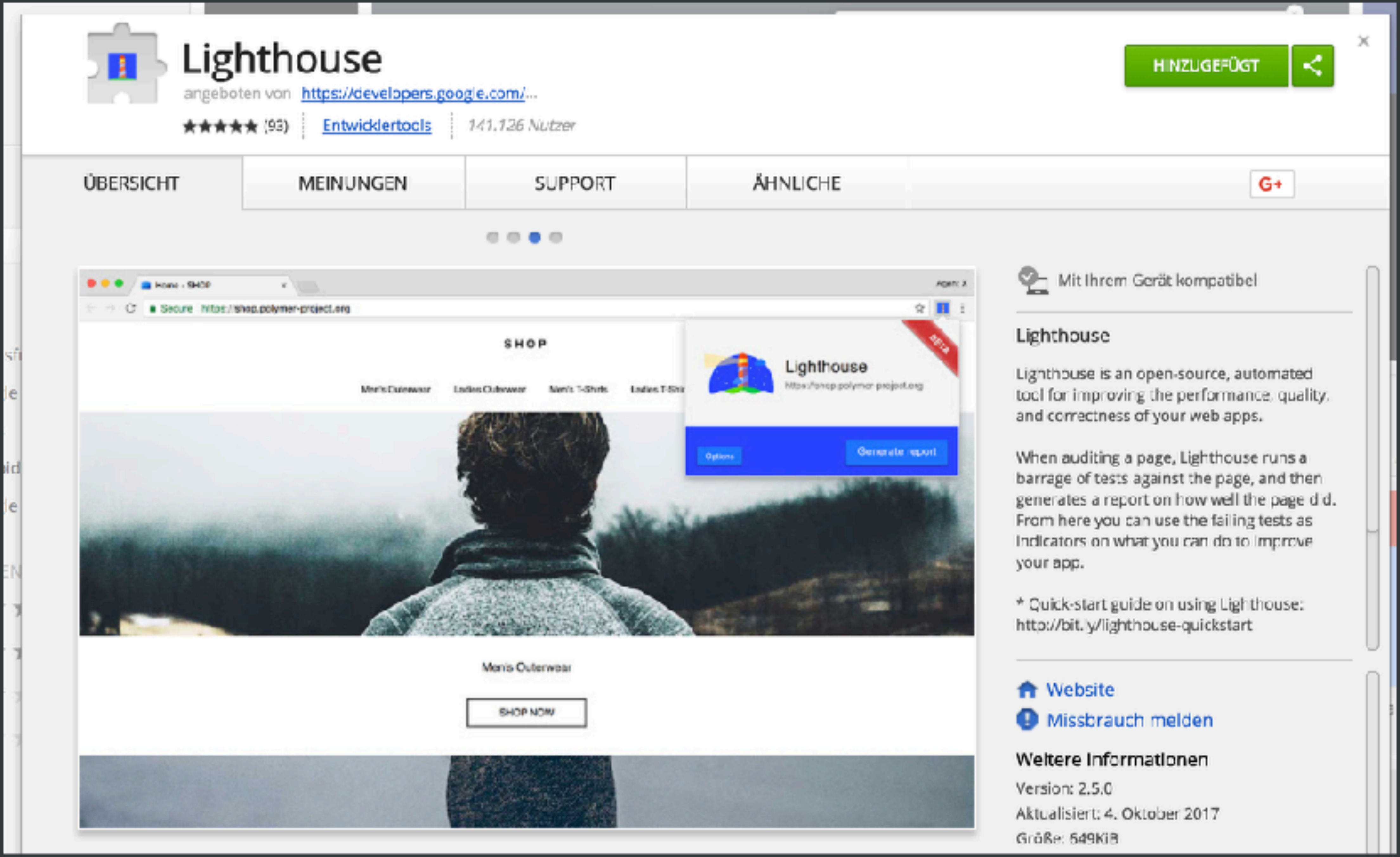
DEVELOP A PWA

IDE

BROWSER : CHROME

LIGHTHOUSE HTTPS://GITHUB.COM/GOOGLECHROME/LIGHTHOUSE

LIGHTHOUSE



Lighthouse

Lighthouse is an open-source, automated tool for improving the performance, quality, and correctness of your web apps.

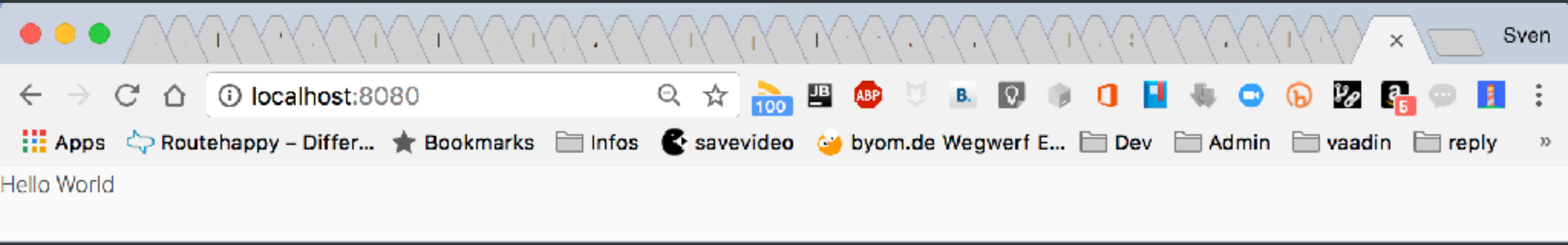
When auditing a page, Lighthouse runs a barrage of tests against the page, and then generates a report on how well the page did. From here you can use the failing tests as indicators on what you can do to improve your app.

* Quick-start guide on using Lighthouse: http://bit.ly/lighthouse-quickstart

Weitere Informationen

Version: 2.5.0
Aktualisiert: 4. Oktober 2017
Größe: 549KiB

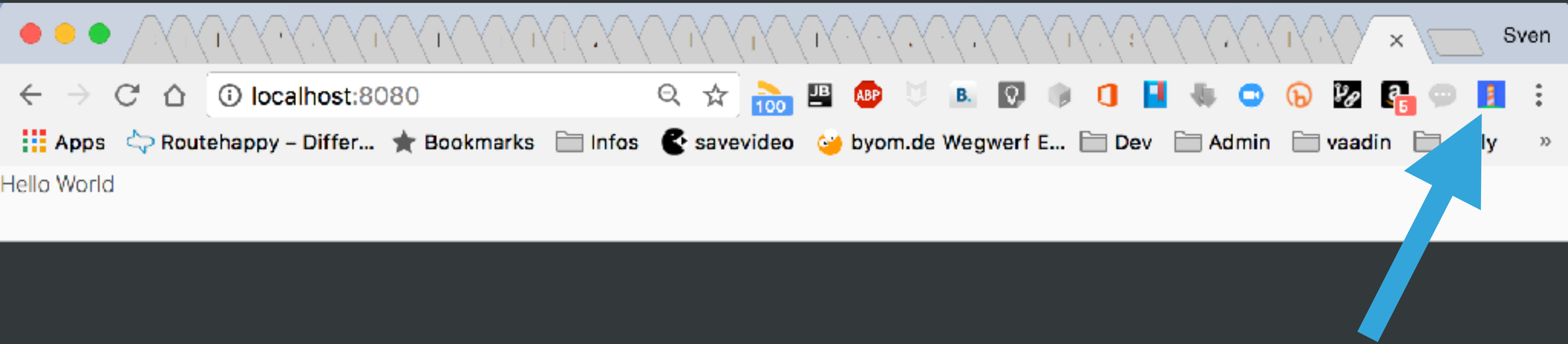
LIGHTHOUSE



Hello World

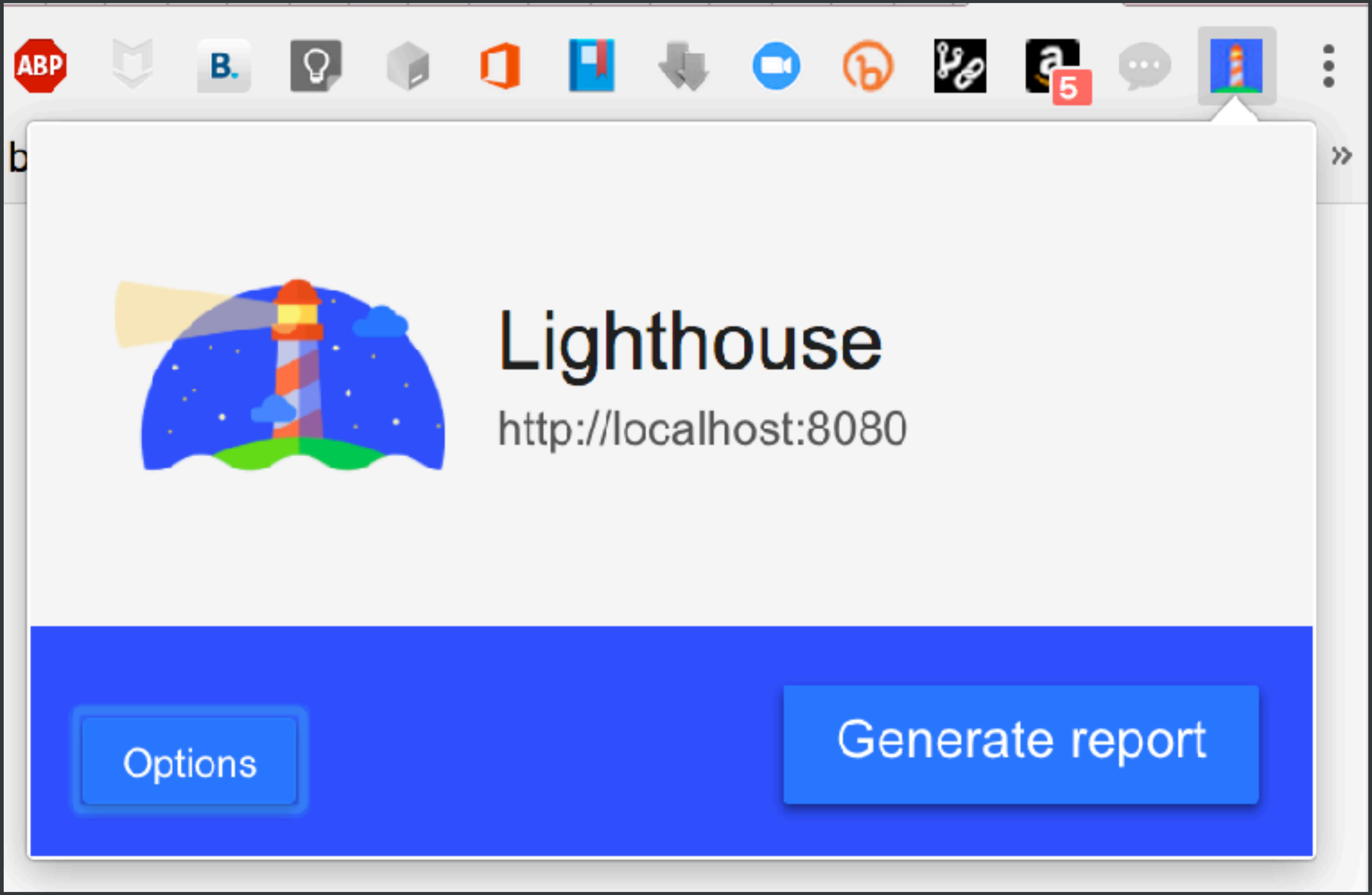
vaadin}>

LIGHTHOUSE



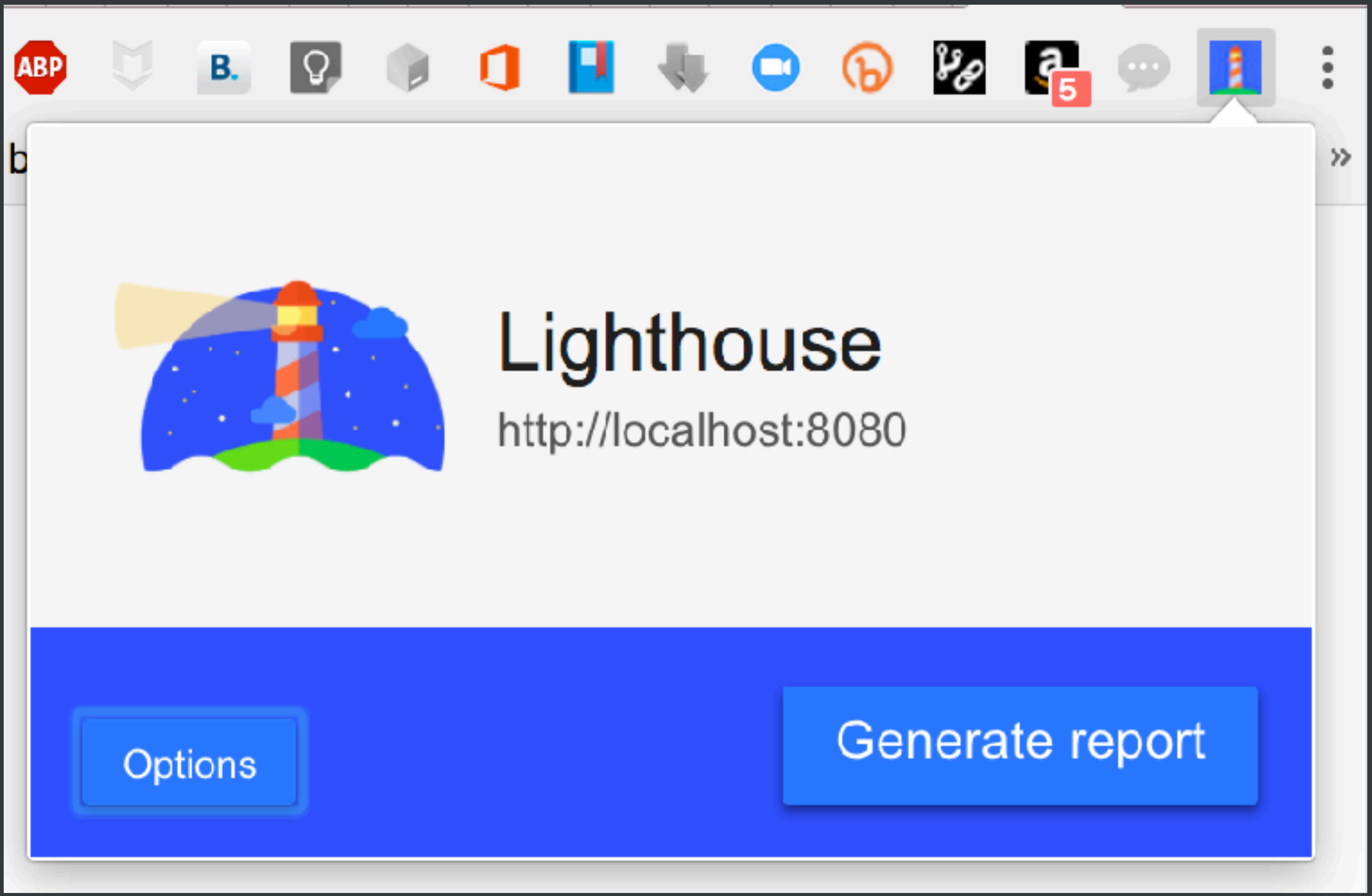
vaadin}>

LIGHTHOUSE



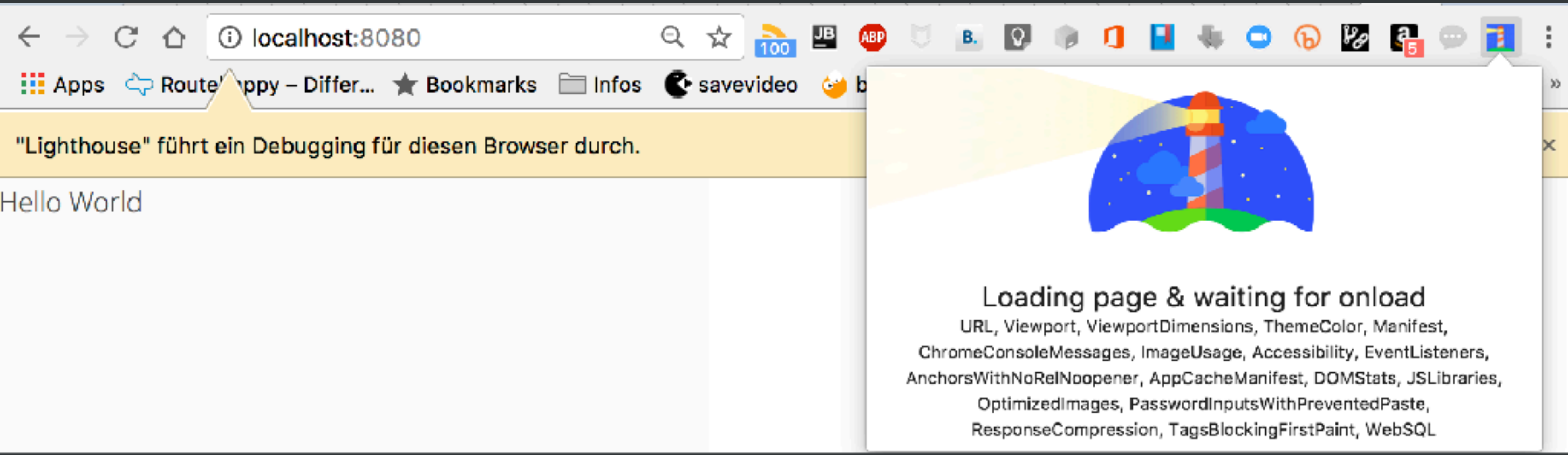
vaadin}>

LIGHTHOUSE



vaadin}>

LIGHTHOUSE



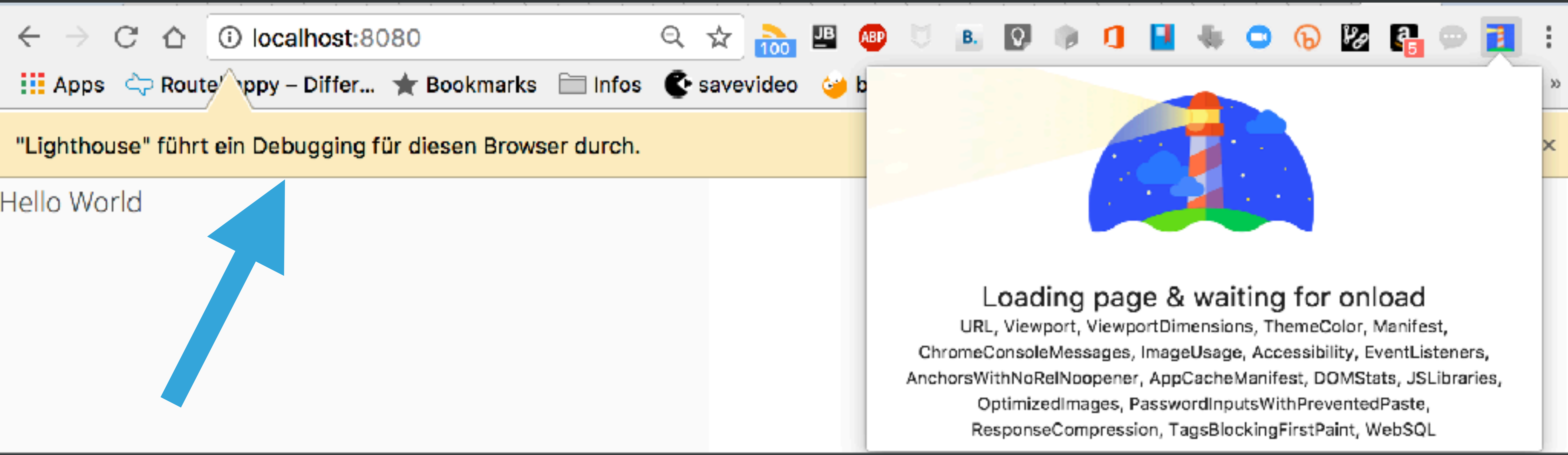
"Lighthouse" führt ein Debugging für diesen Browser durch.

Hello World

Loading page & waiting for onload

URL, Viewport, ViewportDimensions, ThemeColor, Manifest, ChromeConsoleMessages, ImageUsage, Accessibility, EventListeners, AnchorsWithNoRelNoopener, AppCacheManifest, DOMStats, JSLibraries, OptimizedImages, PasswordInputsWithPreventedPaste, ResponseCompression, TagsBlockingFirstPaint, WebSQL

LIGHTHOUSE



"Lighthouse" führt ein Debugging für diesen Browser durch.

Hello World

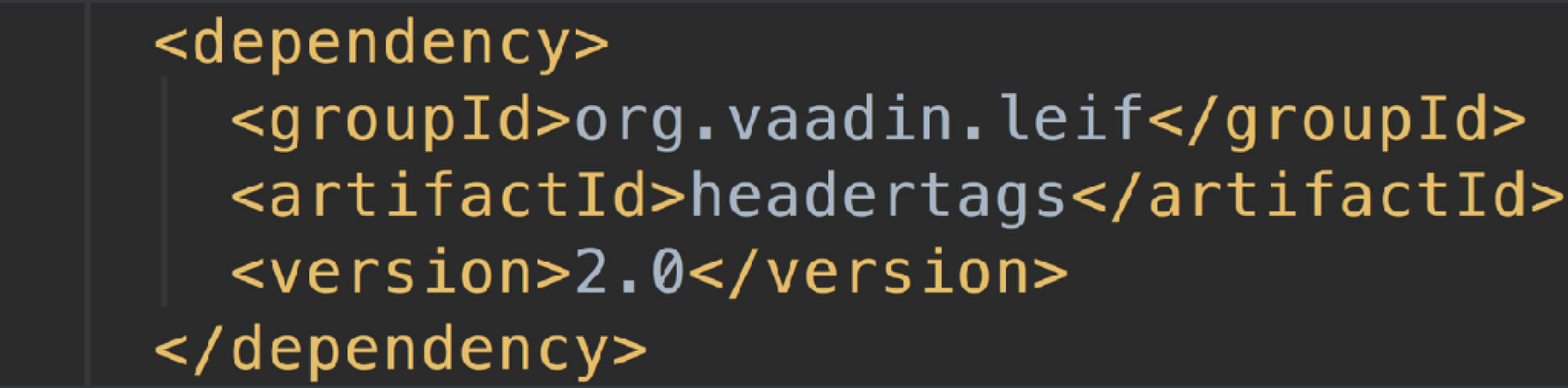
Loading page & waiting for onload

URL, Viewport, ViewportDimensions, ThemeColor, Manifest, ChromeConsoleMessages, ImageUsage, Accessibility, EventListeners, AnchorsWithNoRelNoopener, AppCacheManifest, DOMStats, JSLibraries, OptimizedImages, PasswordInputsWithPreventedPaste, ResponseCompression, TagsBlockingFirstPaint, WebSQL

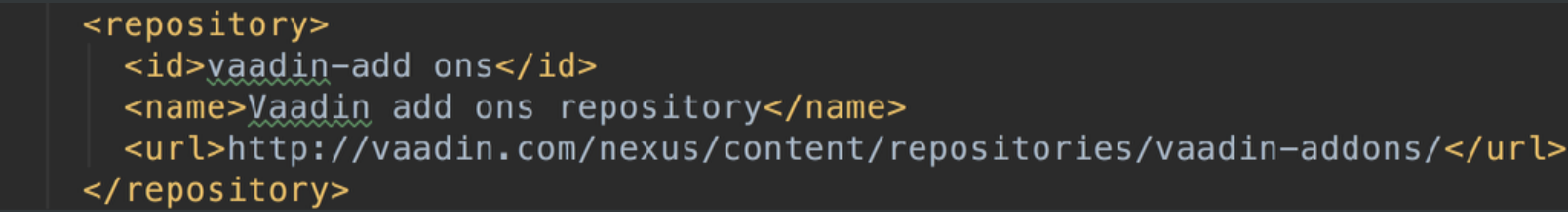
vaadin}>

CONNECT WITH VAADIN

CONNECT WITH VAADIN

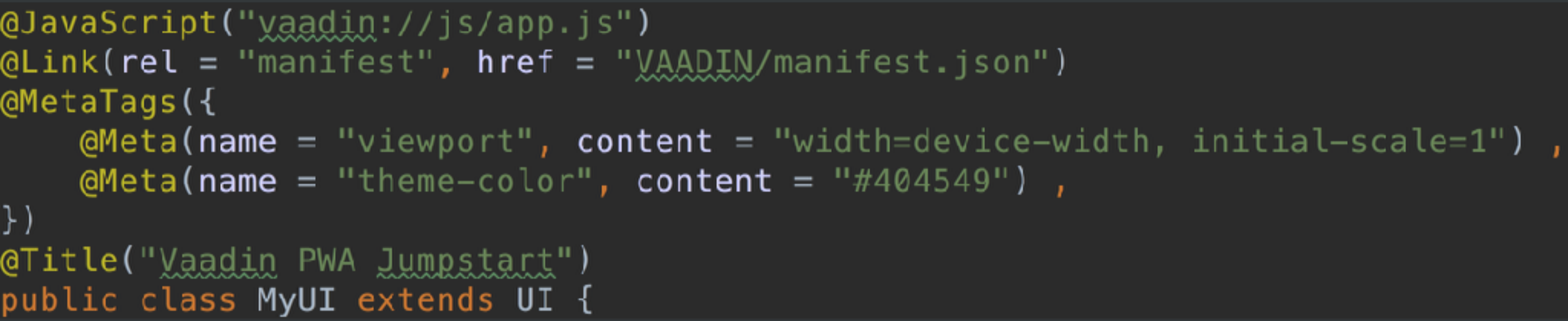


```xml
<dependency>
  <groupId>org.vaadin.leif</groupId>
  <artifactId>headertags</artifactId>
  <version>2.0</version>
</dependency>
```



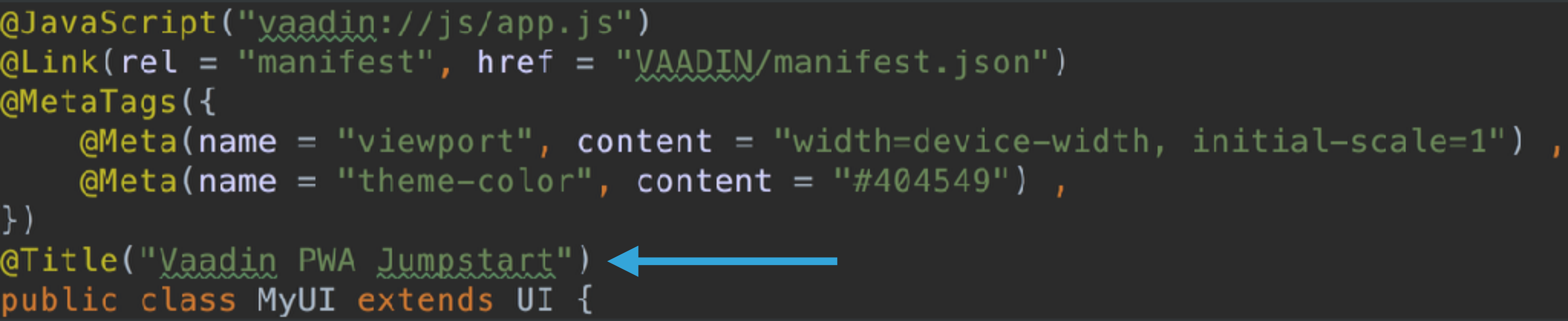
```xml
<repository>
  <id>vaadin-add ons</id>
  <name>Vaadin add ons repository</name>
  <url>http://vaadin.com/nexus/content/repositories/vaadin-addons/</url>
</repository>
```

CONNECT WITH VAADIN



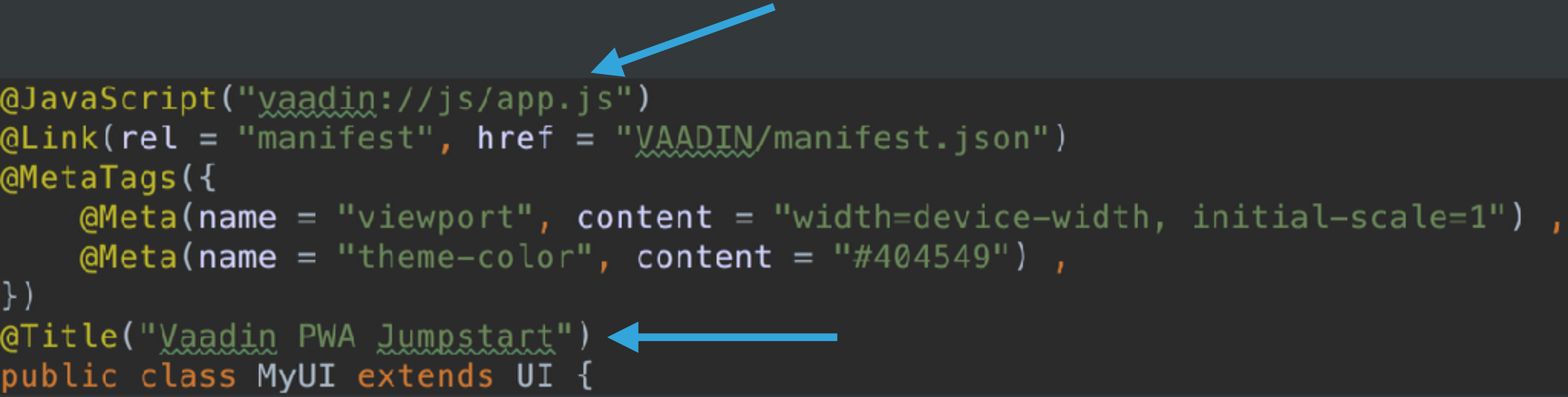
```
@JavaScript("vaadin://js/app.js")
@Link(rel = "manifest", href = "VAADIN/manifest.json")
@MetaTags({
    @Meta(name = "viewport", content = "width=device-width, initial-scale=1") ,
    @Meta(name = "theme-color", content = "#404549") ,
})
@Title("Vaadin PWA Jumpstart")
public class MyUI extends UI {
```

CONNECT WITH VAADIN



```
@JavaScript("vaadin://js/app.js")
@Link(rel = "manifest", href = "VAADIN/manifest.json")
@MetaTags({
    @Meta(name = "viewport", content = "width=device-width, initial-scale=1") ,
    @Meta(name = "theme-color", content = "#404549") ,
})
@Title("Vaadin PWA Jumpstart")
public class MyUI extends UI {
```

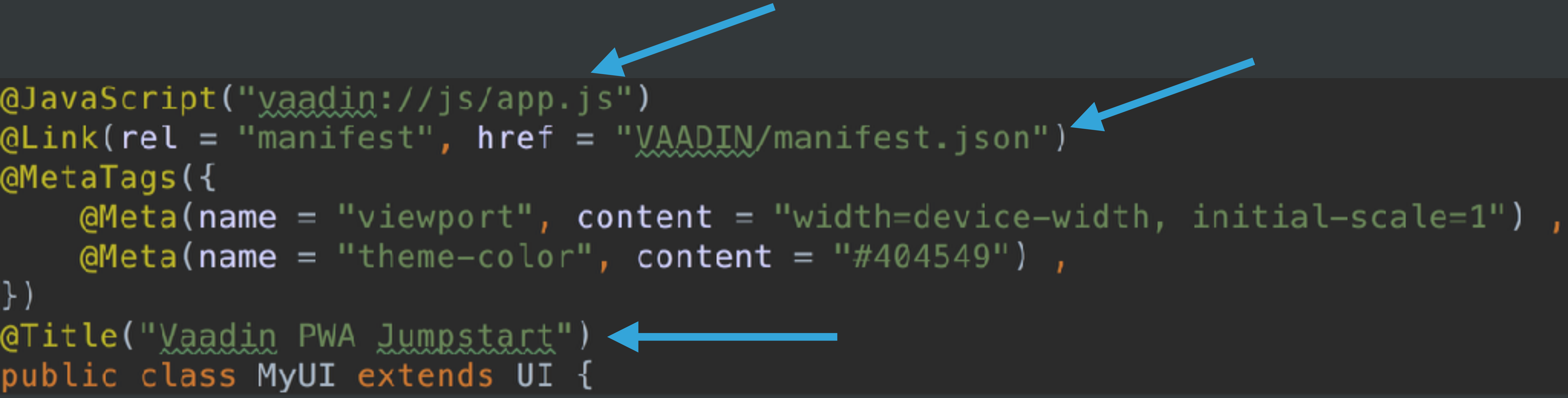
CONNECT WITH VAADIN



```
@JavaScript("vaadin://js/app.js")
@Link(rel = "manifest", href = "VAADIN/manifest.json")
@MetaTags({
    @Meta(name = "viewport", content = "width=device-width, initial-scale=1") ,
    @Meta(name = "theme-color", content = "#404549") ,
})
@Title("Vaadin PWA Jumpstart")
public class MyUI extends UI {
```

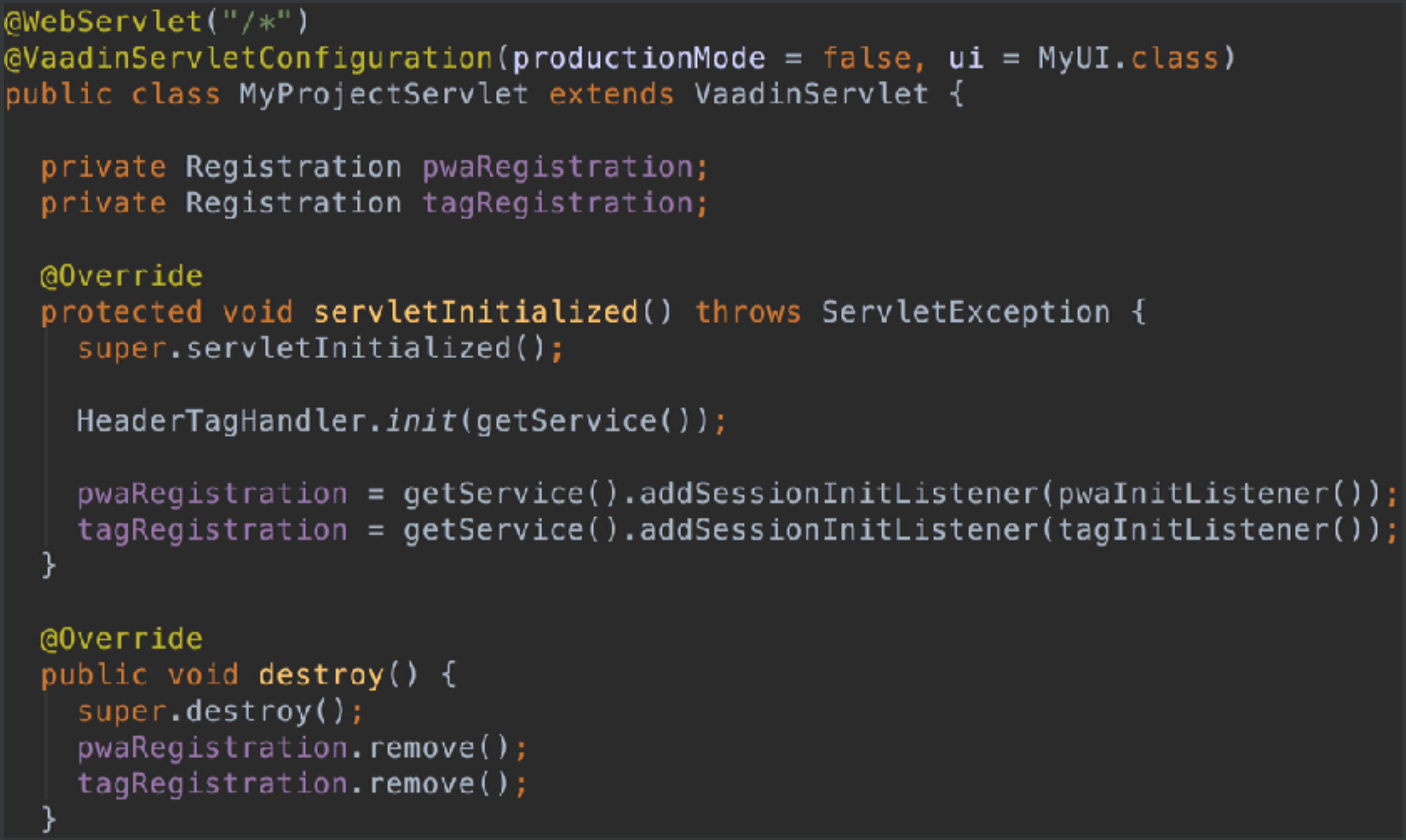
vaadin}>

CONNECT WITH VAADIN



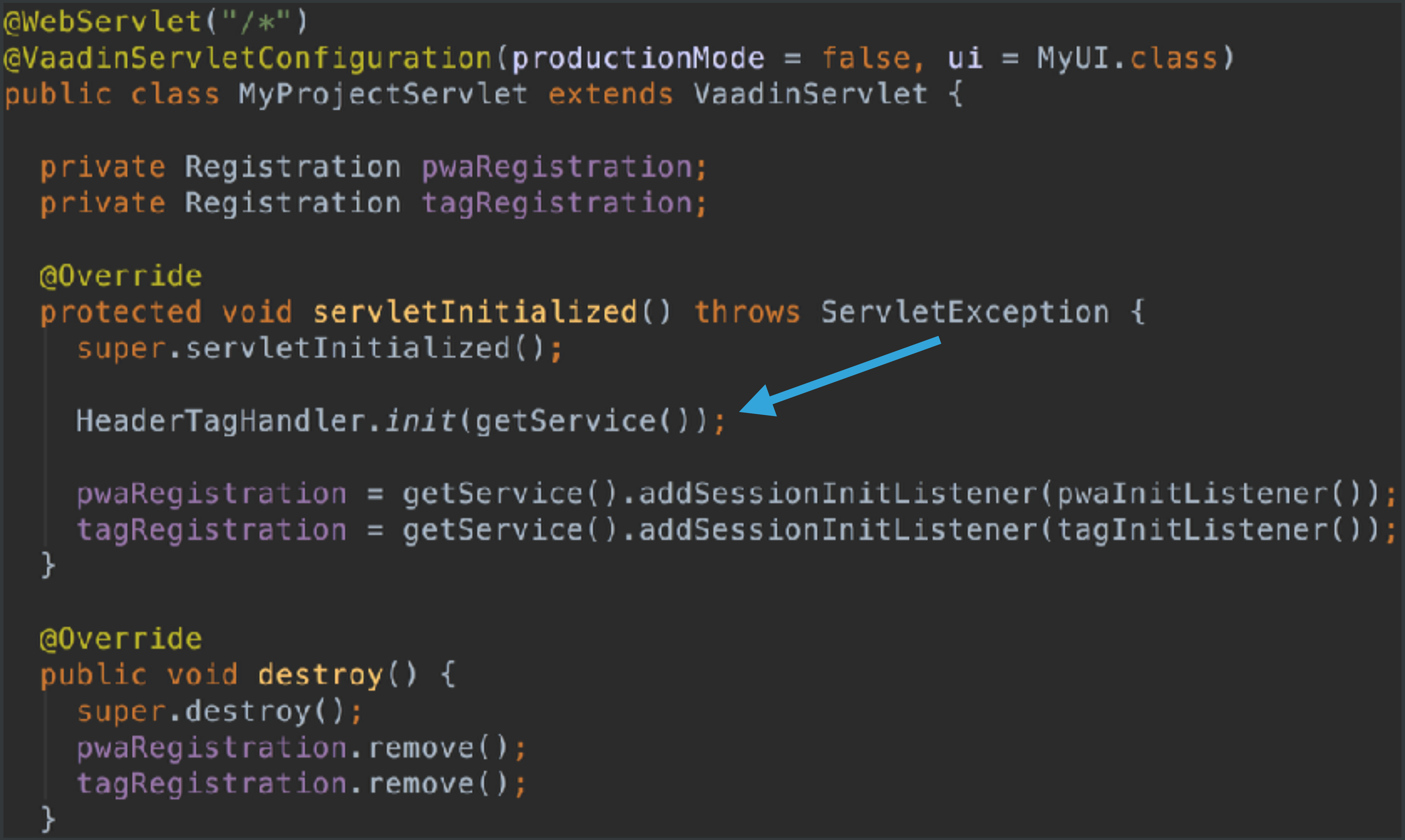
```java
@JavaScript("vaadin://js/app.js")
@Link(rel = "manifest", href = "VAADIN/manifest.json")
@MetaTags({
    @Meta(name = "viewport", content = "width=device-width, initial-scale=1") ,
    @Meta(name = "theme-color", content = "#404549") ,
})
@Title("Vaadin PWA Jumpstart")
public class MyUI extends UI {
```

CONNECT WITH VAADIN



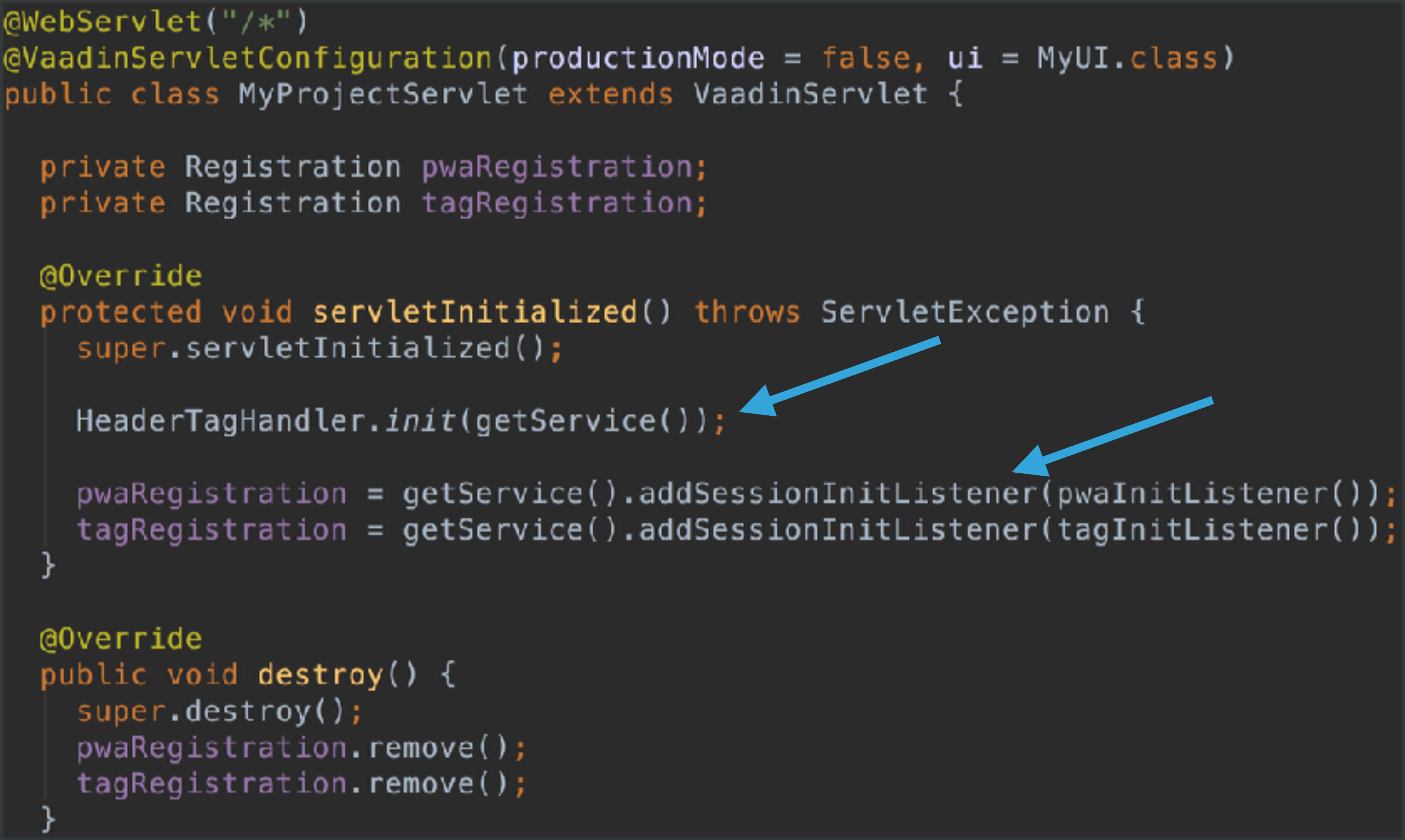
```
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {

  private Registration pwaRegistration;
  private Registration tagRegistration;

  @Override
  protected void servletInitialized() throws ServletException {
    super.servletInitialized();

    HeaderTagHandler.init(getService());

    pwaRegistration = getService().addSessionInitListener(pwaInitListener());
    tagRegistration = getService().addSessionInitListener(tagInitListener());
  }

  @Override
  public void destroy() {
    super.destroy();
    pwaRegistration.remove();
    tagRegistration.remove();
  }
```

CONNECT WITH VAADIN



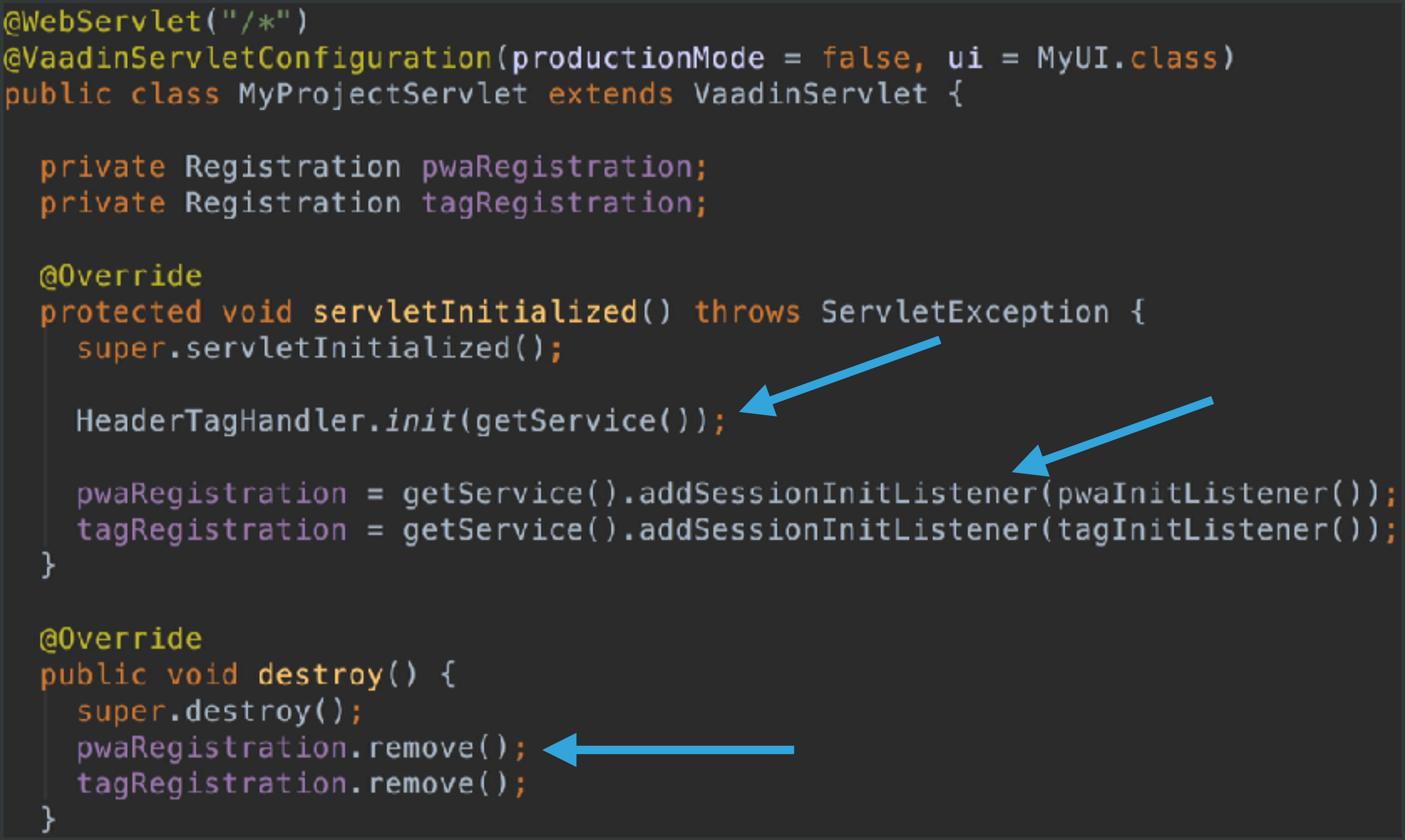
```java
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {

  private Registration pwaRegistration;
  private Registration tagRegistration;

  @Override
  protected void servletInitialized() throws ServletException {
    super.servletInitialized();

    HeaderTagHandler.init(getService());

    pwaRegistration = getService().addSessionInitListener(pwaInitListener());
    tagRegistration = getService().addSessionInitListener(tagInitListener());
  }

  @Override
  public void destroy() {
    super.destroy();
    pwaRegistration.remove();
    tagRegistration.remove();
  }
```

CONNECT WITH VAADIN



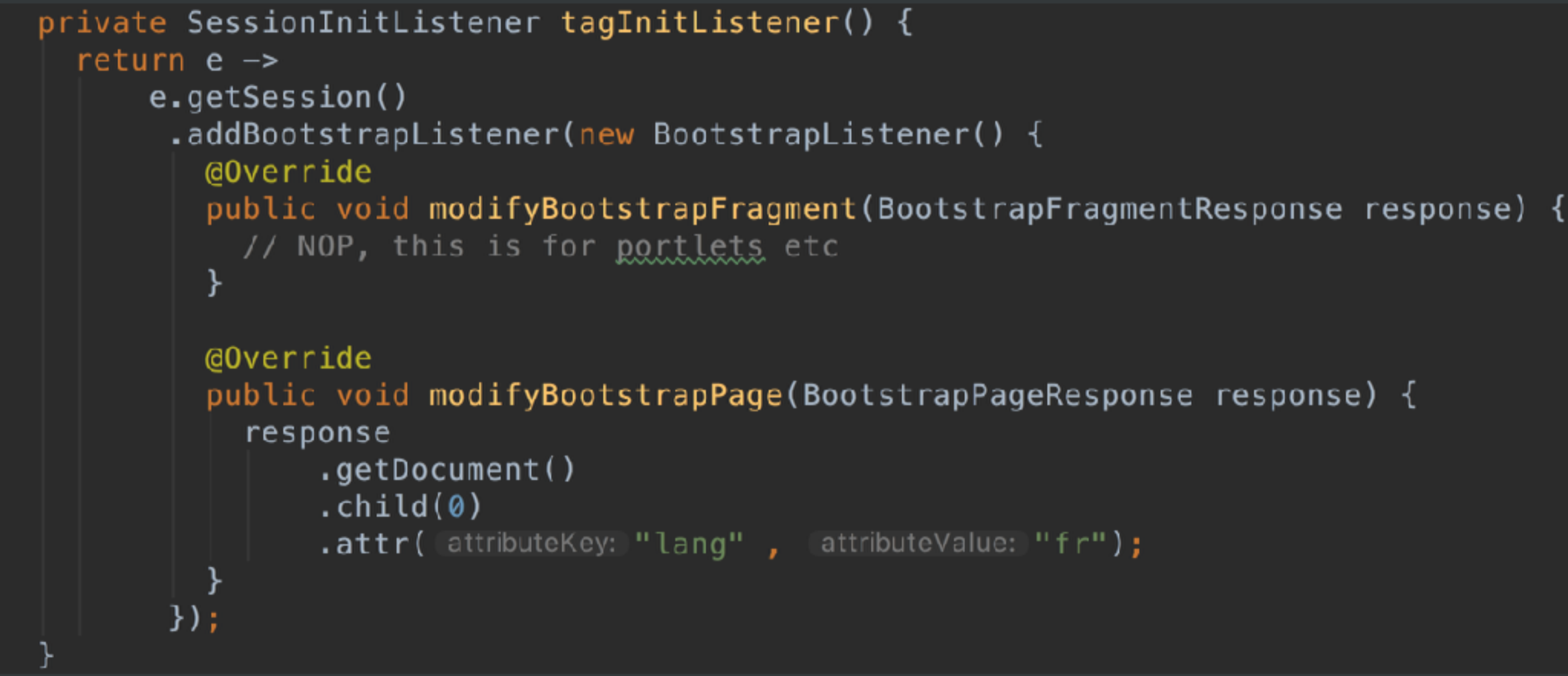
```java
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {

  private Registration pwaRegistration;
  private Registration tagRegistration;

  @Override
  protected void servletInitialized() throws ServletException {
    super.servletInitialized();

    HeaderTagHandler.init(getService());

    pwaRegistration = getService().addSessionInitListener(pwaInitListener());
    tagRegistration = getService().addSessionInitListener(tagInitListener());
  }

  @Override
  public void destroy() {
    super.destroy();
    pwaRegistration.remove();
    tagRegistration.remove();
  }
```

CONNECT WITH VAADIN



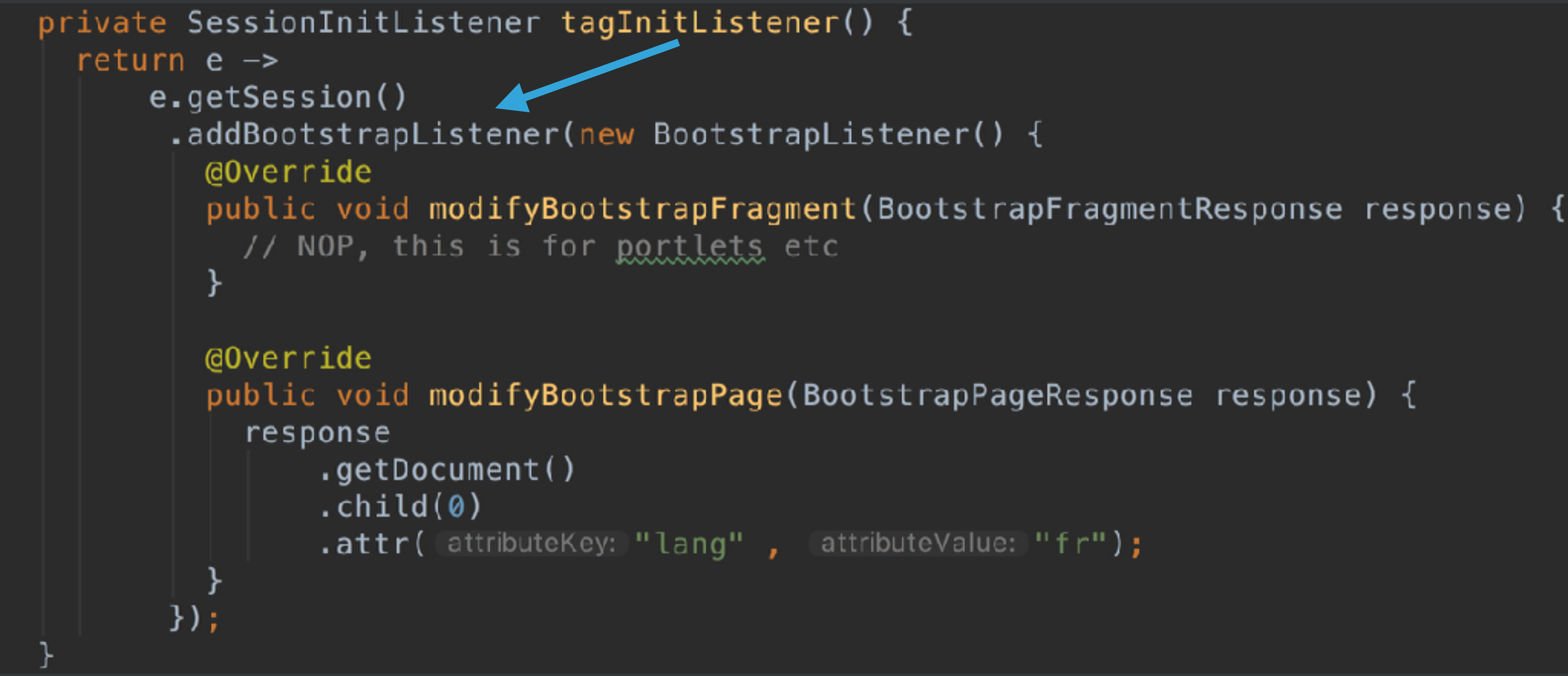
```java
@WebServlet("/*")
@VaadinServletConfiguration(productionMode = false, ui = MyUI.class)
public class MyProjectServlet extends VaadinServlet {

  private Registration pwaRegistration;
  private Registration tagRegistration;

  @Override
  protected void servletInitialized() throws ServletException {
    super.servletInitialized();

    HeaderTagHandler.init(getService());

    pwaRegistration = getService().addSessionInitListener(pwaInitListener());
    tagRegistration = getService().addSessionInitListener(tagInitListener());
  }

  @Override
  public void destroy() {
    super.destroy();
    pwaRegistration.remove();
    tagRegistration.remove();
  }
```

CONNECT WITH VAADIN



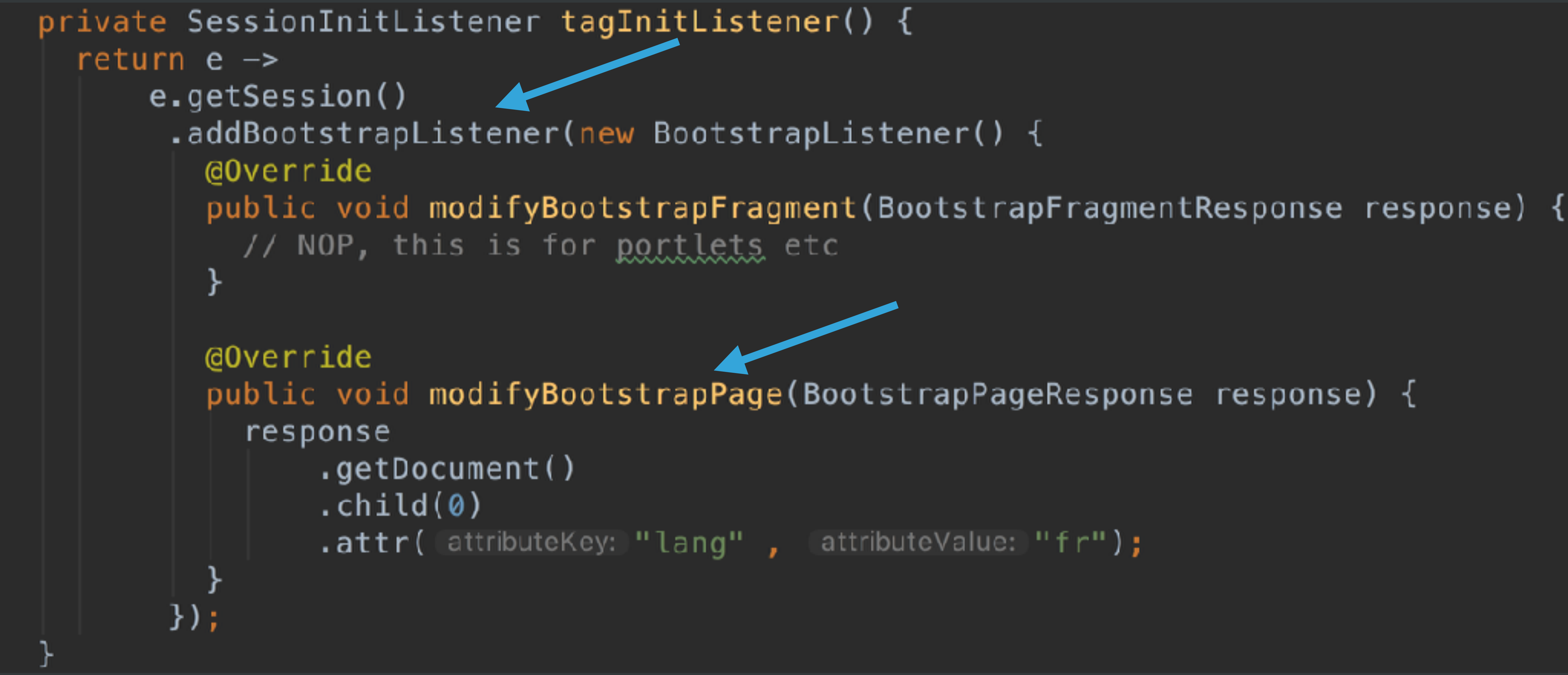
```java
private SessionInitListener tagInitListener() {
  return e ->
      e.getSession()
       .addBootstrapListener(new BootstrapListener() {
         @Override
         public void modifyBootstrapFragment(BootstrapFragmentResponse response) {
           // NOP, this is for portlets etc
         }

         @Override
         public void modifyBootstrapPage(BootstrapPageResponse response) {
           response
               .getDocument()
               .child(0)
               .attr( attributeKey: "lang" , attributeValue: "fr");
         }
       });
}
```

CONNECT WITH VAADIN



```
private SessionInitListener tagInitListener() {
  return e ->
      e.getSession()
       .addBootstrapListener(new BootstrapListener() {
         @Override
         public void modifyBootstrapFragment(BootstrapFragmentResponse response) {
           // NOP, this is for portlets etc
         }

         @Override
         public void modifyBootstrapPage(BootstrapPageResponse response) {
           response
               .getDocument()
               .child(0)
               .attr( attributeKey: "lang" ,  attributeValue: "fr");
         }
       });
}
```

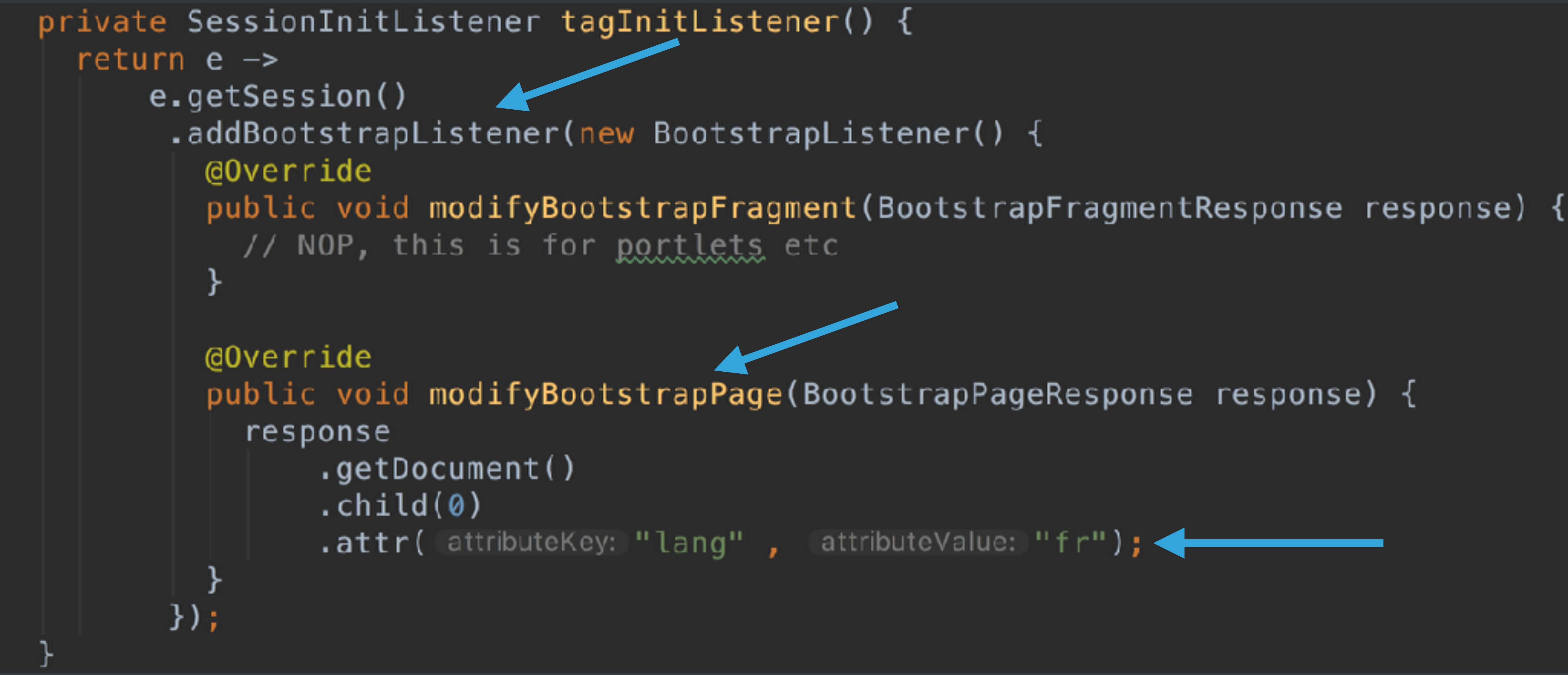
CONNECT WITH VAADIN



```java
private SessionInitListener tagInitListener() {
  return e ->
      e.getSession()
       .addBootstrapListener(new BootstrapListener() {
         @Override
         public void modifyBootstrapFragment(BootstrapFragmentResponse response) {
           // NOP, this is for portlets etc
         }

         @Override
         public void modifyBootstrapPage(BootstrapPageResponse response) {
           response
               .getDocument()
               .child(0)
               .attr( attributeKey: "lang" ,  attributeValue: "fr");
         }
       });
}
```

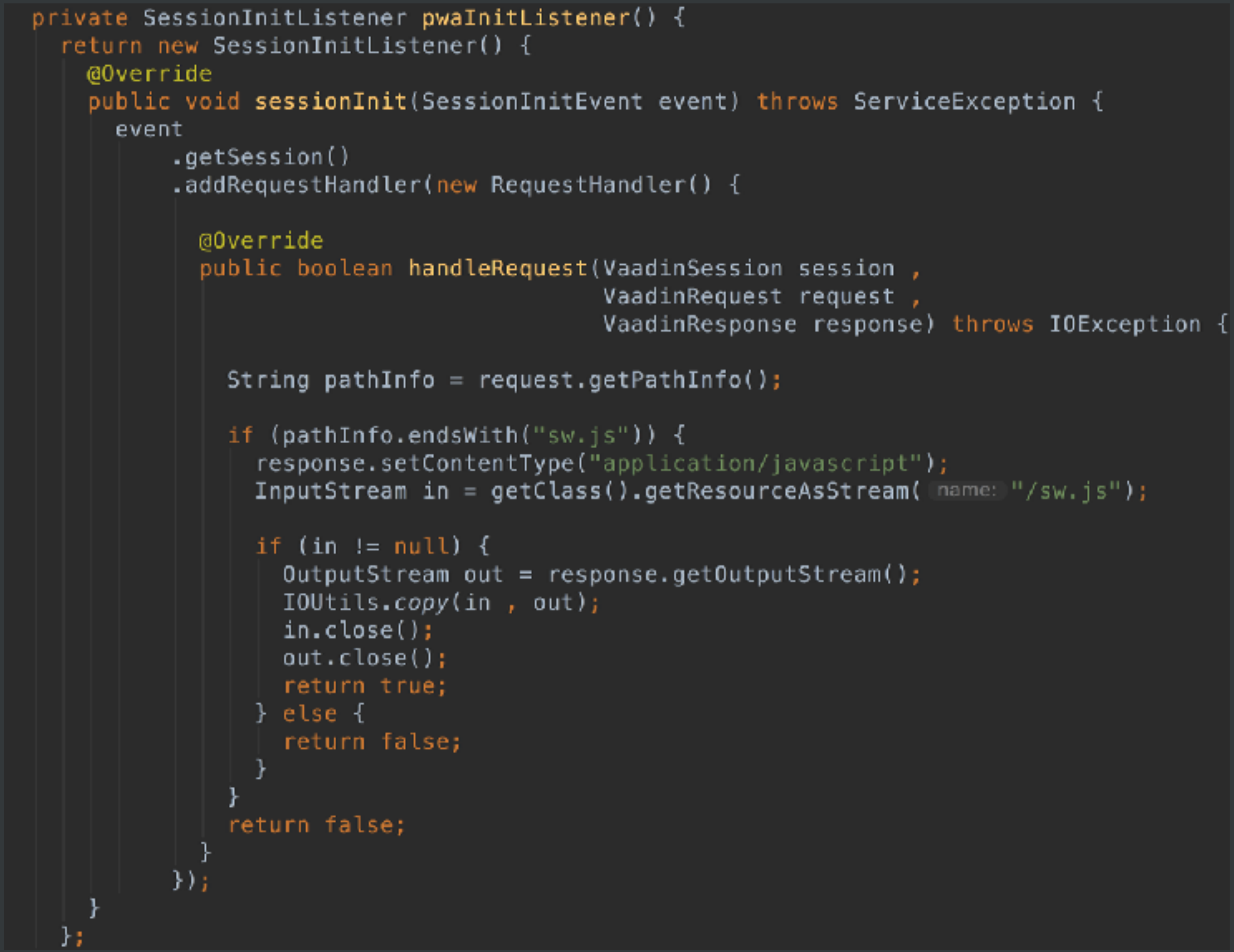
vaadin}>

CONNECT WITH VAADIN



```
private SessionInitListener tagInitListener() {
  return e ->
      e.getSession()
       .addBootstrapListener(new BootstrapListener() {
         @Override
         public void modifyBootstrapFragment(BootstrapFragmentResponse response) {
           // NOP, this is for portlets etc
         }

         @Override
         public void modifyBootstrapPage(BootstrapPageResponse response) {
           response
               .getDocument()
               .child(0)
               .attr( attributeKey: "lang" ,  attributeValue: "fr");
         }
       });
}
```

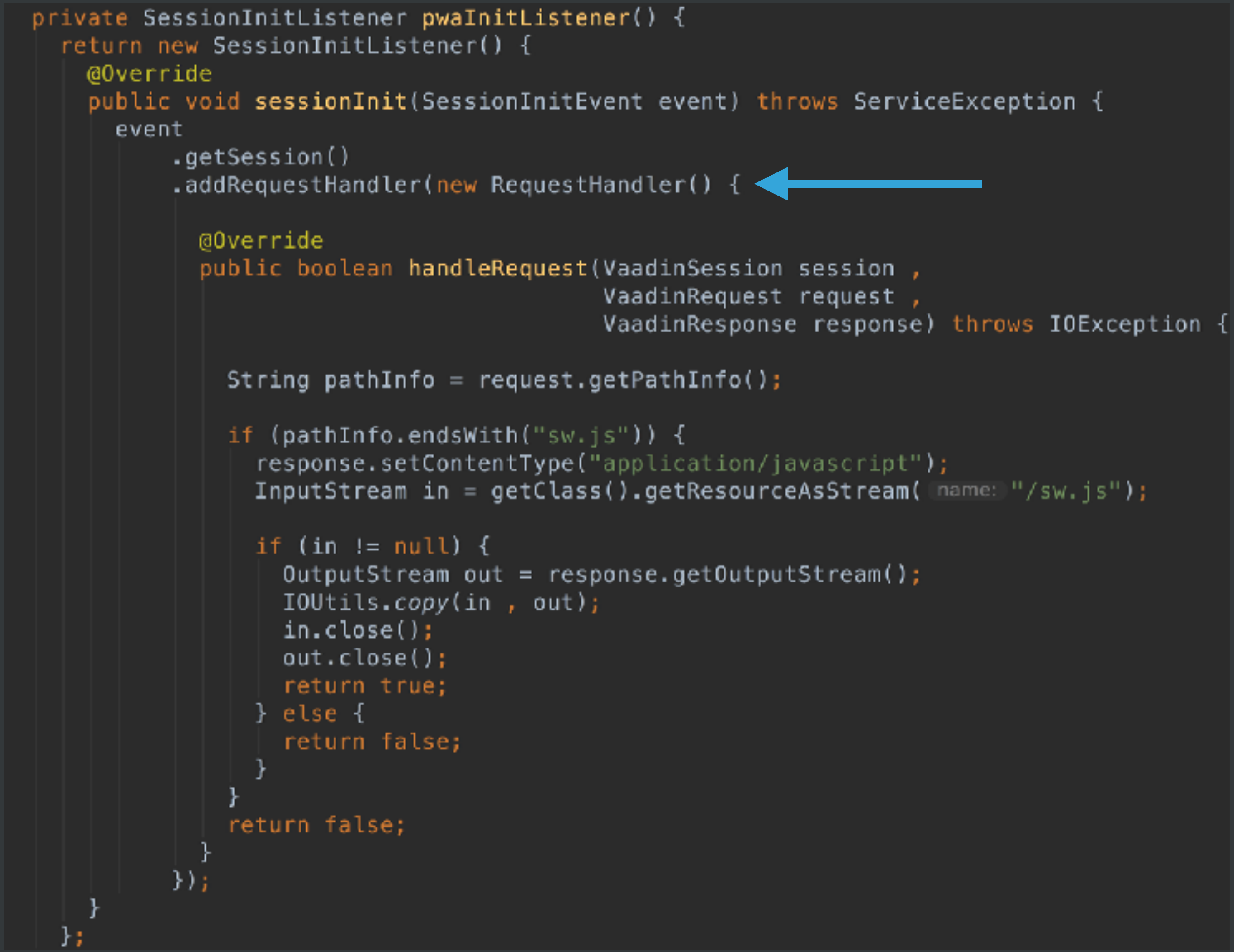
CONNECT WITH VAADIN



```java
private SessionInitListener pwaInitListener() {
  return new SessionInitListener() {
    @Override
    public void sessionInit(SessionInitEvent event) throws ServiceException {
      event
          .getSession()
          .addRequestHandler(new RequestHandler() {

            @Override
            public boolean handleRequest(VaadinSession session ,
                                         VaadinRequest request ,
                                         VaadinResponse response) throws IOException {

              String pathInfo = request.getPathInfo();

              if (pathInfo.endsWith("sw.js")) {
                response.setContentType("application/javascript");
                InputStream in = getClass().getResourceAsStream( name: "/sw.js");

                if (in != null) {
                  OutputStream out = response.getOutputStream();
                  IOUtils.copy(in , out);
                  in.close();
                  out.close();
                  return true;
                } else {
                  return false;
                }
              }
              return false;
            }
          });
    }
  };
```

vaadin}>

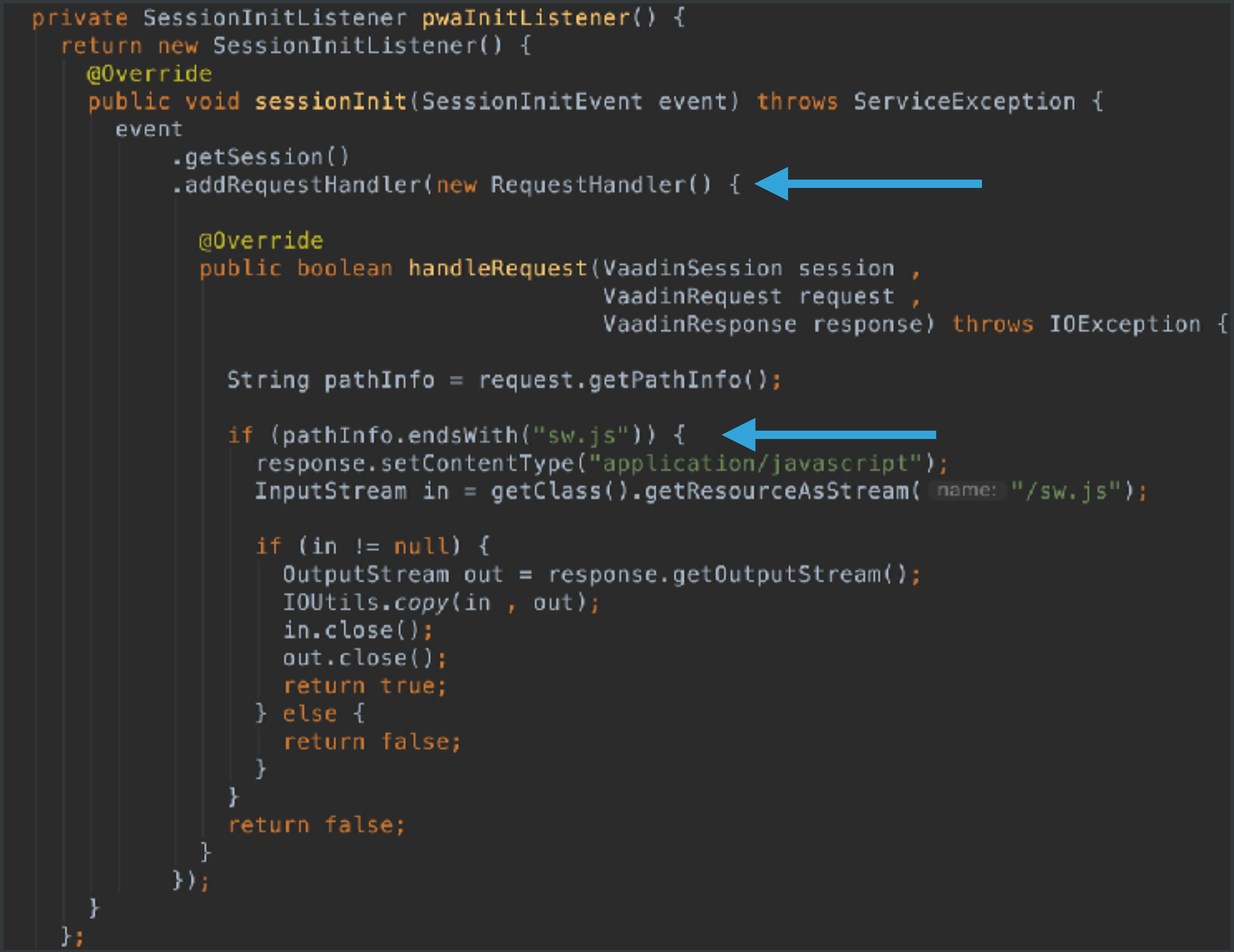
CONNECT WITH VAADIN



```
private SessionInitListener pwaInitListener() {
  return new SessionInitListener() {
    @Override
    public void sessionInit(SessionInitEvent event) throws ServiceException {
      event
          .getSession()
          .addRequestHandler(new RequestHandler() {

            @Override
            public boolean handleRequest(VaadinSession session ,
                                         VaadinRequest request ,
                                         VaadinResponse response) throws IOException {

              String pathInfo = request.getPathInfo();

              if (pathInfo.endsWith("sw.js")) {
                response.setContentType("application/javascript");
                InputStream in = getClass().getResourceAsStream( name: "/sw.js");

                if (in != null) {
                  OutputStream out = response.getOutputStream();
                  IOUtils.copy(in , out);
                  in.close();
                  out.close();
                  return true;
                } else {
                  return false;
                }
              }
              return false;
            }
          });
    }
  };
```

vaadin}>

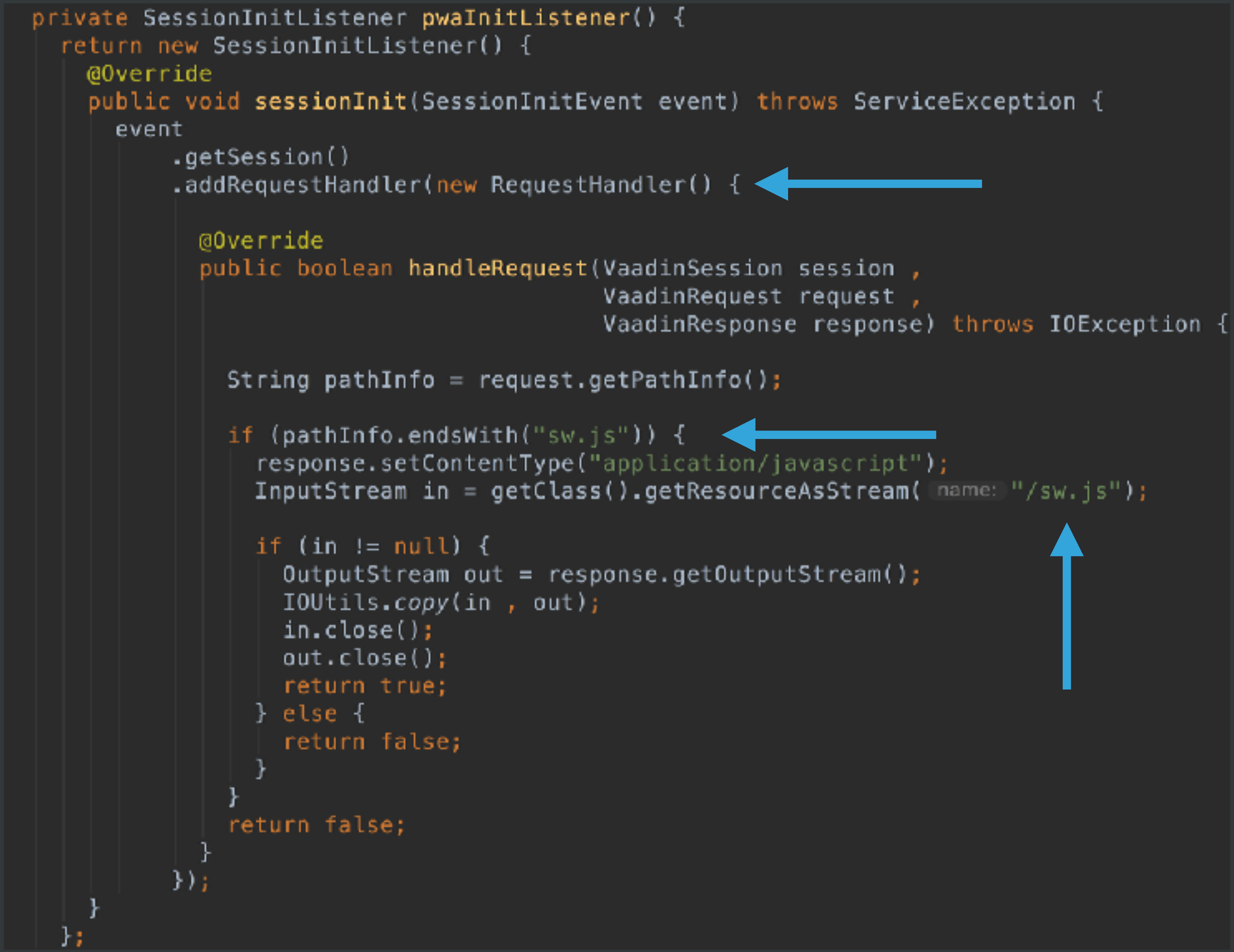
CONNECT WITH VAADIN



```java
private SessionInitListener pwaInitListener() {
  return new SessionInitListener() {
    @Override
    public void sessionInit(SessionInitEvent event) throws ServiceException {
      event
          .getSession()
          .addRequestHandler(new RequestHandler() {

            @Override
            public boolean handleRequest(VaadinSession session ,
                                         VaadinRequest request ,
                                         VaadinResponse response) throws IOException {

              String pathInfo = request.getPathInfo();

              if (pathInfo.endsWith("sw.js")) {
                response.setContentType("application/javascript");
                InputStream in = getClass().getResourceAsStream( name: "/sw.js");

                if (in != null) {
                  OutputStream out = response.getOutputStream();
                  IOUtils.copy(in , out);
                  in.close();
                  out.close();
                  return true;
                } else {
                  return false;
                }
              }
              return false;
            }
          });
    }
  };
```

vaadin}>

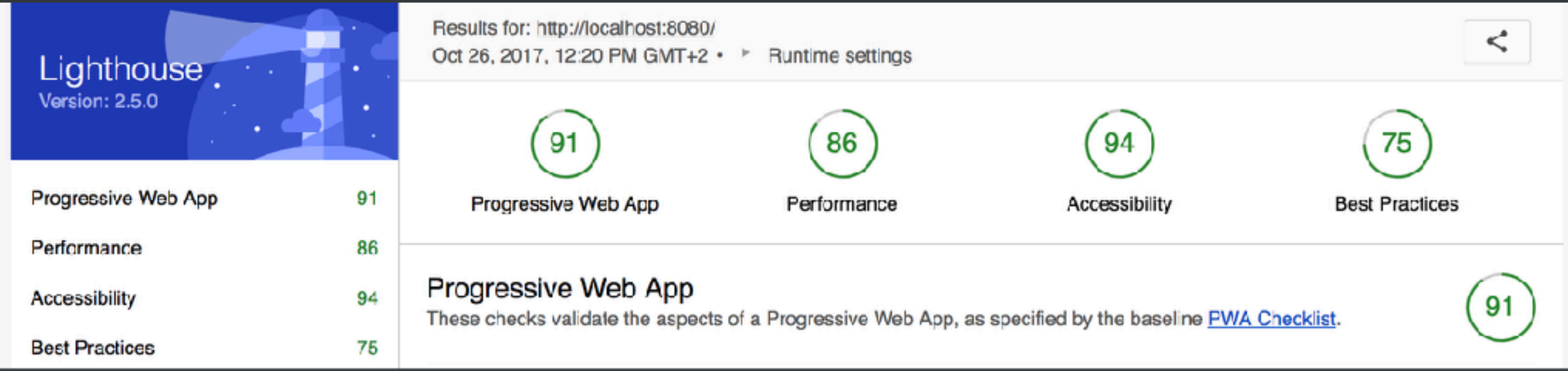
CONNECT WITH VAADIN



```
private SessionInitListener pwaInitListener() {
  return new SessionInitListener() {
    @Override
    public void sessionInit(SessionInitEvent event) throws ServiceException {
      event
          .getSession()
          .addRequestHandler(new RequestHandler() {

            @Override
            public boolean handleRequest(VaadinSession session ,
                                         VaadinRequest request ,
                                         VaadinResponse response) throws IOException {

              String pathInfo = request.getPathInfo();

              if (pathInfo.endsWith("sw.js")) {
                response.setContentType("application/javascript");
                InputStream in = getClass().getResourceAsStream( name: "/sw.js");

                if (in != null) {
                  OutputStream out = response.getOutputStream();
                  IOUtils.copy(in , out);
                  in.close();
                  out.close();
                  return true;
                } else {
                  return false;
                }
              }
              return false;
            }
          });
    }
  };
```

vaadin}>

CONNECT WITH VAADIN



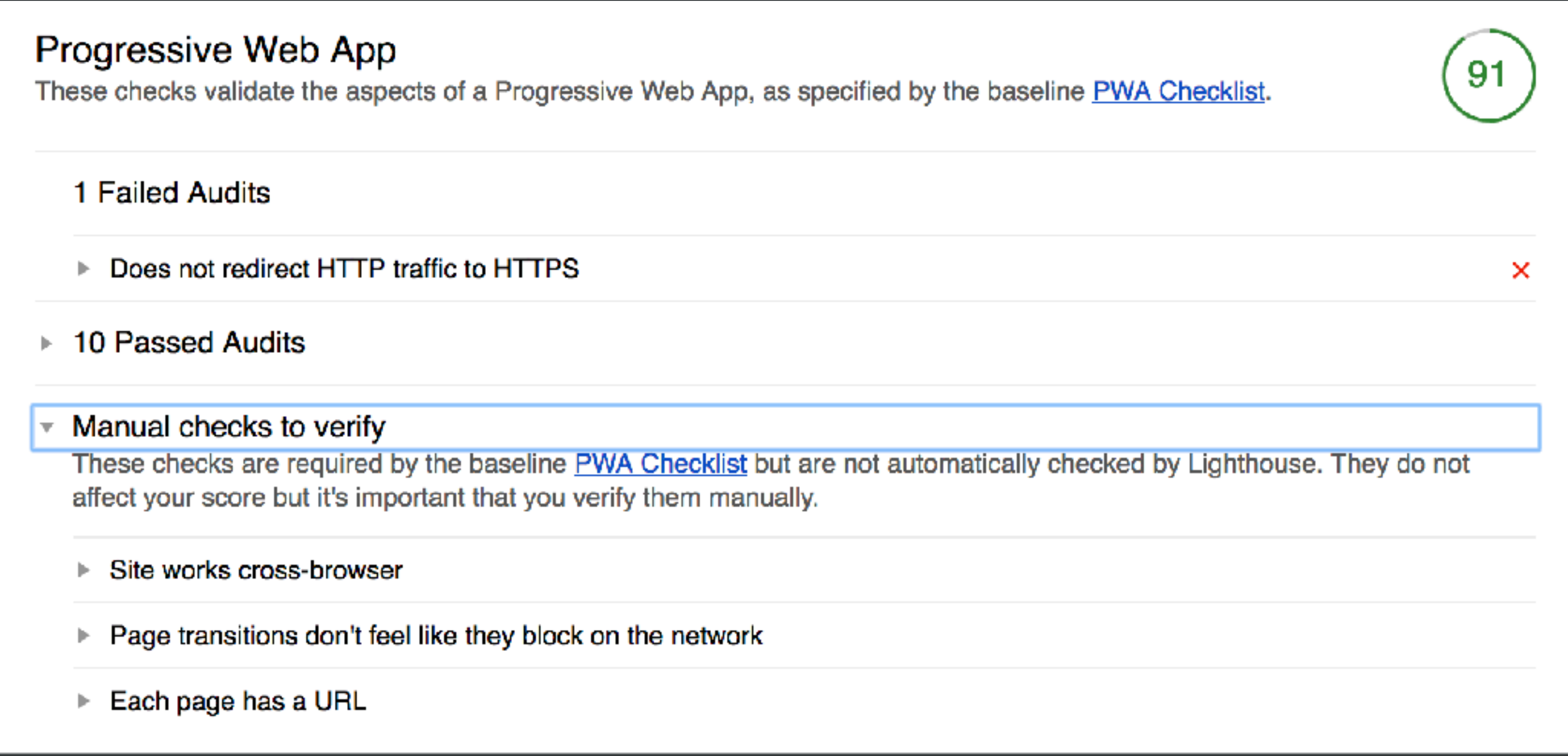
Results for: http://localhost:8080/
Oct 26, 2017, 12:20 PM GMT+2 • Runtime settings

Progressive Web App	91
Performance	86
Accessibility	94
Best Practices	75

Progressive Web App
These checks validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist. 91

vaadin}>

CONNECT WITH VAADIN



Progressive Web App

These checks validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.

91

1 Failed Audits

▸ Does not redirect HTTP traffic to HTTPS ✗

▸ 10 Passed Audits

▾ Manual checks to verify

These checks are required by the baseline PWA Checklist but are not automatically checked by Lighthouse. They do not affect your score but it's important that you verify them manually.

▸ Site works cross-browser

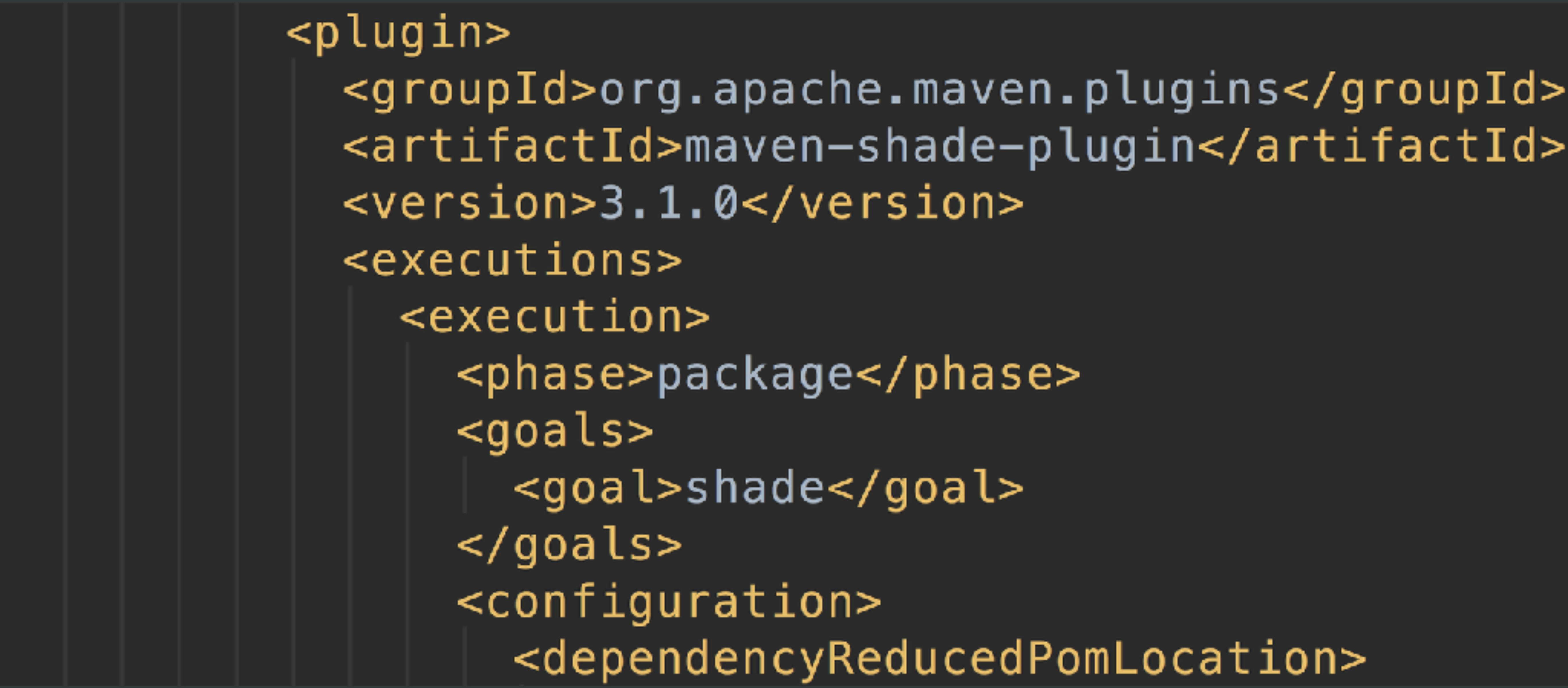
▸ Page transitions don't feel like they block on the network

▸ Each page has a URL

vaadin}>

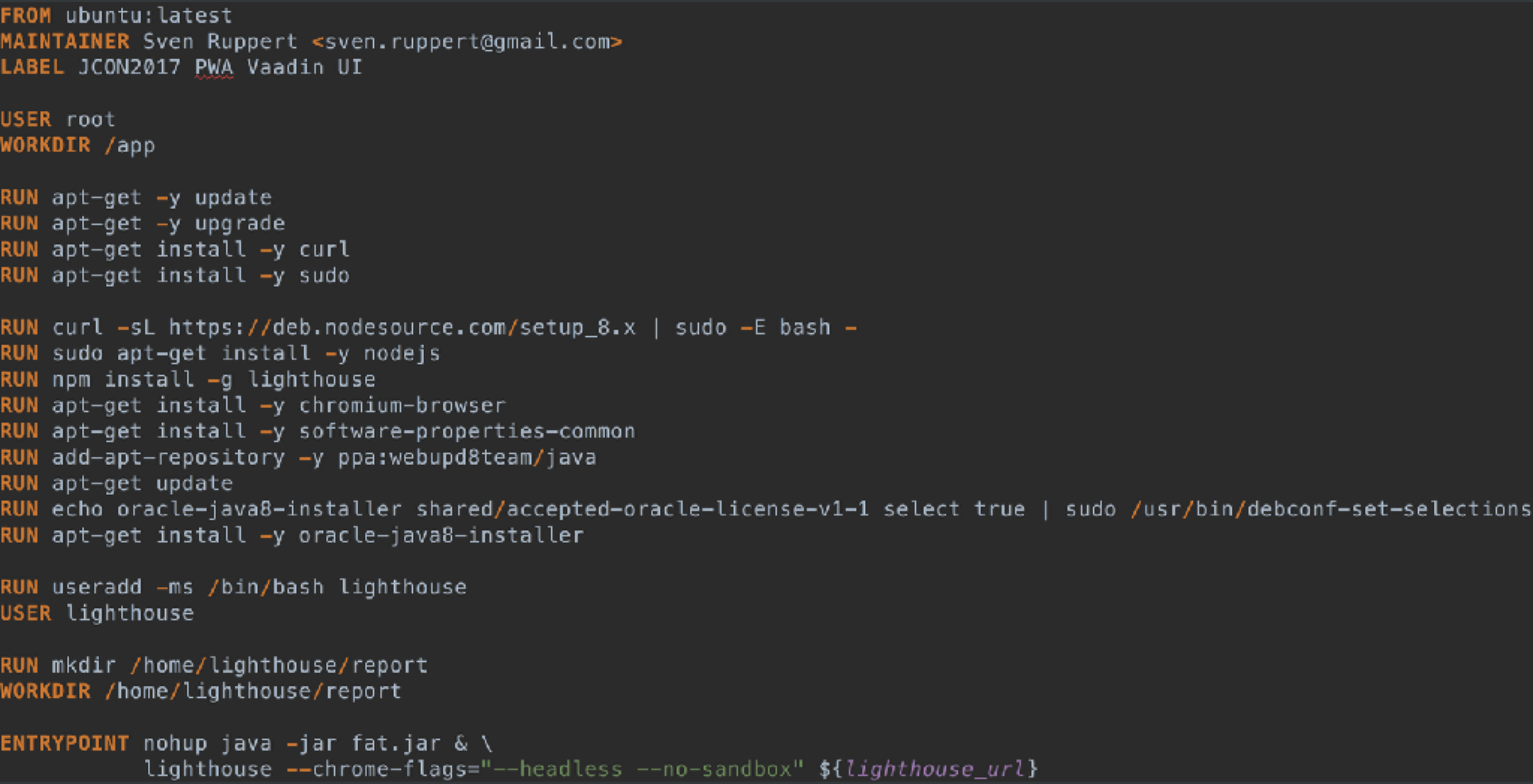
GO DOCKER

GO DOCKER



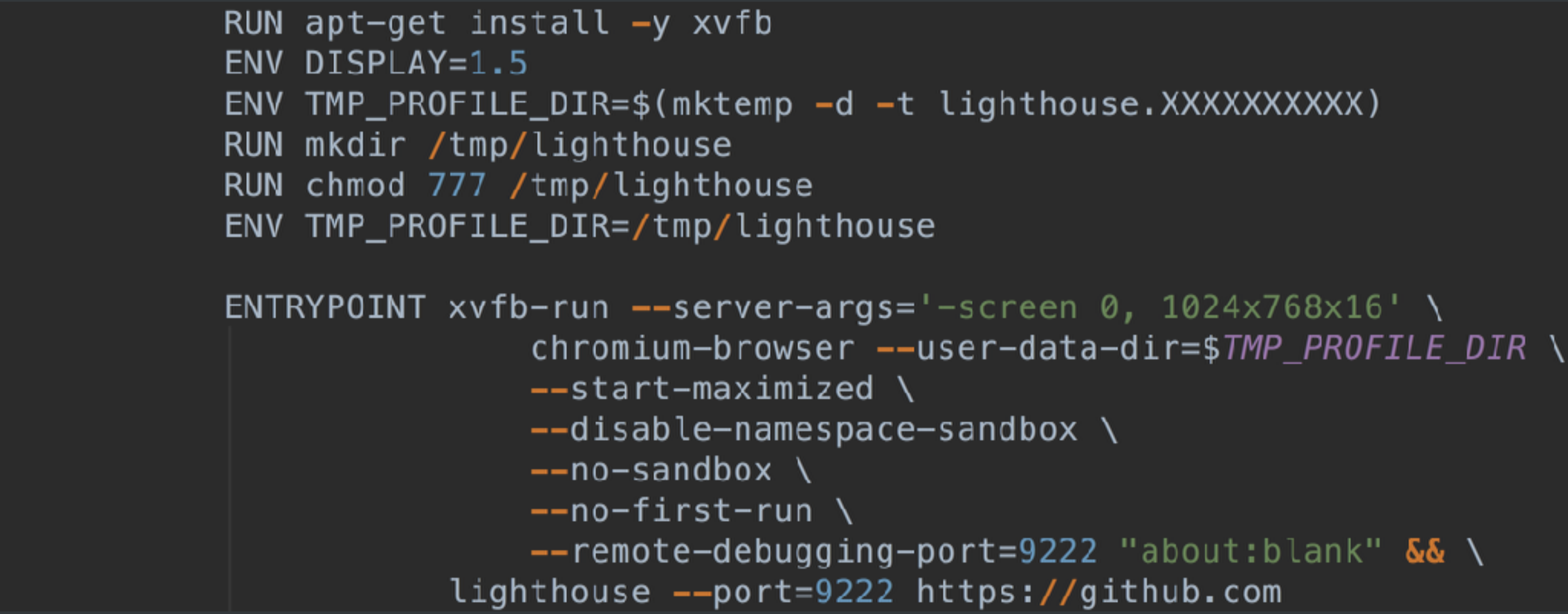
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>3.1.0</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <dependencyReducedPomLocation>
```

GO DOCKER



```dockerfile
FROM ubuntu:latest
MAINTAINER Sven Ruppert <sven.ruppert@gmail.com>
LABEL JCON2017 PWA Vaadin UI

USER root
WORKDIR /app

RUN apt-get -y update
RUN apt-get -y upgrade
RUN apt-get install -y curl
RUN apt-get install -y sudo

RUN curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
RUN sudo apt-get install -y nodejs
RUN npm install -g lighthouse
RUN apt-get install -y chromium-browser
RUN apt-get install -y software-properties-common
RUN add-apt-repository -y ppa:webupd8team/java
RUN apt-get update
RUN echo oracle-java8-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
RUN apt-get install -y oracle-java8-installer

RUN useradd -ms /bin/bash lighthouse
USER lighthouse

RUN mkdir /home/lighthouse/report
WORKDIR /home/lighthouse/report

ENTRYPOINT nohup java -jar fat.jar & \
           lighthouse --chrome-flags="--headless --no-sandbox" ${lighthouse_url}
```

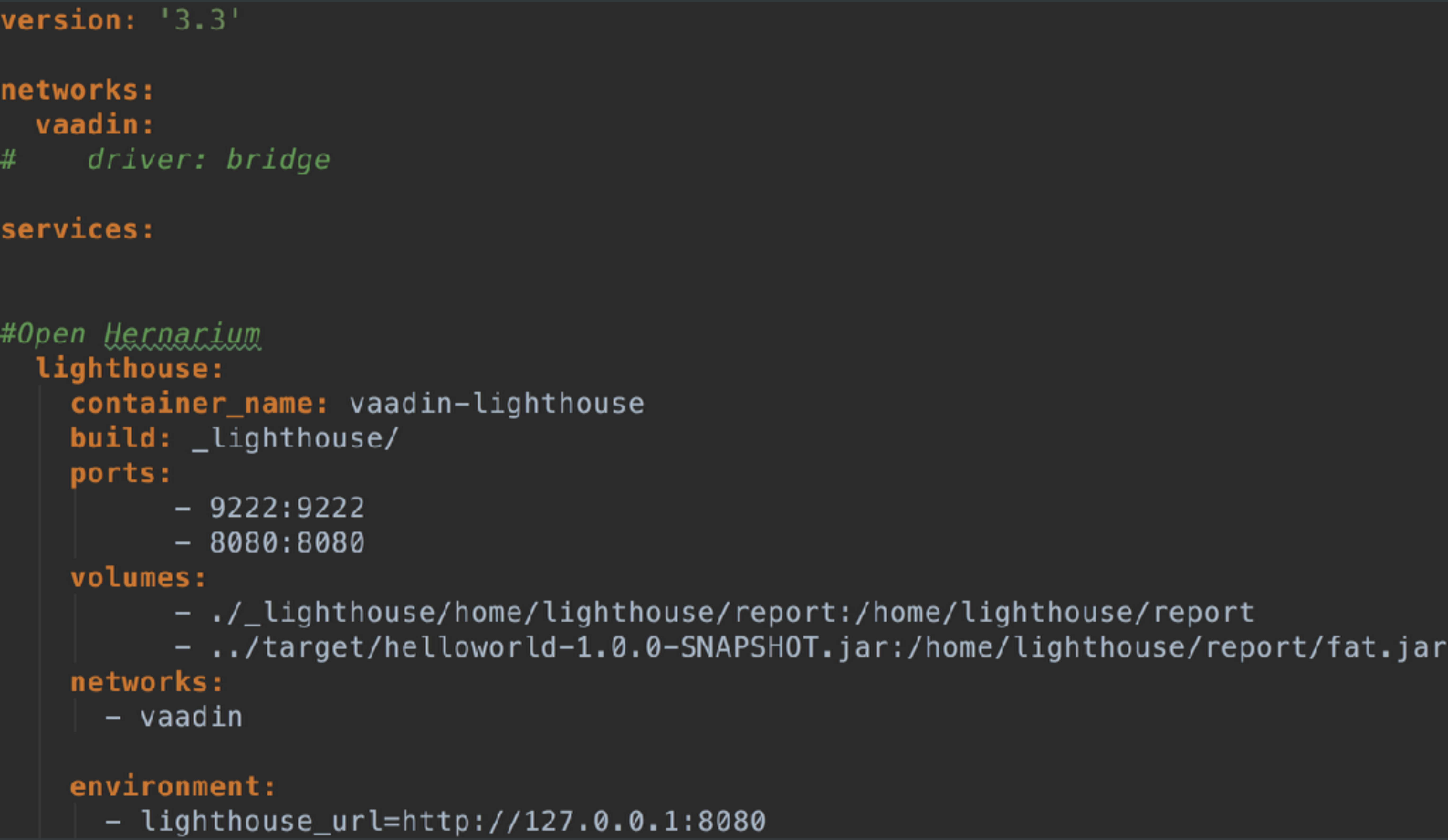
GO DOCKER



```
RUN apt-get install -y xvfb
ENV DISPLAY=1.5
ENV TMP_PROFILE_DIR=$(mktemp -d -t lighthouse.XXXXXXXXXX)
RUN mkdir /tmp/lighthouse
RUN chmod 777 /tmp/lighthouse
ENV TMP_PROFILE_DIR=/tmp/lighthouse

ENTRYPOINT xvfb-run --server-args='-screen 0, 1024x768x16' \
               chromium-browser --user-data-dir=$TMP_PROFILE_DIR \
               --start-maximized \
               --disable-namespace-sandbox \
               --no-sandbox \
               --no-first-run \
               --remote-debugging-port=9222 "about:blank" && \
           lighthouse --port=9222 https://github.com
```

vaadin}>

GO DOCKER



```yaml
version: '3.3'

networks:
  vaadin:
#     driver: bridge

services:


#Open Hernarium
  lighthouse:
    container_name: vaadin-lighthouse
    build: _lighthouse/
    ports:
          - 9222:9222
          - 8080:8080
    volumes:
          - ./_lighthouse/home/lighthouse/report:/home/lighthouse/report
          - ../target/helloworld-1.0.0-SNAPSHOT.jar:/home/lighthouse/report/fat.jar
    networks:
      - vaadin

    environment:
      - lighthouse_url=http://127.0.0.1:8080
```

vaadin}>

THANK YOU

@SVENRUPPERT

vaadin}>

T-SHIRT!

PAGES.VAADIN.COM/GET-TSHIRT

vaadin}>