

# Web Programming

# **Login**

# Login

- This lecture: Simple login using sessions.
  - Has some security flaws
- Deployment alternatives:
  - Flask-Login
  - OAuth provider, e.g. firebase.google.com

# **Accessibility**

# Accessibility

- In Norway, public websites must be universally accessible
- People with limited hearing, sight must be able to use the page
- Requirements follow the accessibility standard from the W3C  
WACG: <https://www.w3.org/WAI/standards-guidelines/>

# Accessibility - Design

- Text must be readable
- Text must be adjustable:
  - Font size
  - Line height
- Contrast between colors
- Usable with large zoom
- ...

# Accessibility - Keyboard

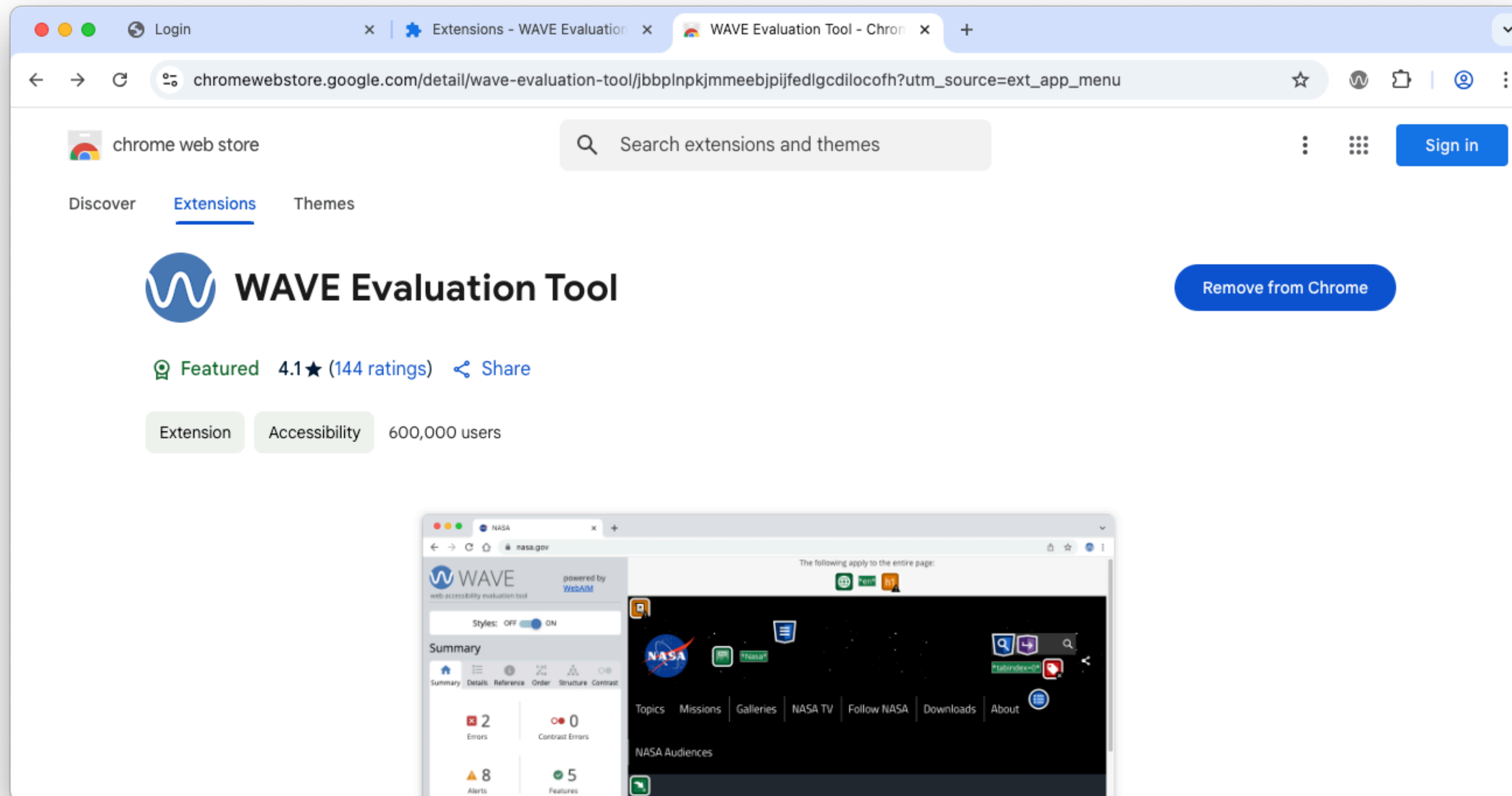
- Page usable with keyboard only
- No hidden buttons
  - Use button for elements with onclick
- Avoid complex mouse interaction
- Manage focus

```
document.getElementById("mod_name").focus();
```

# Accessibility - Screen reader

- Use semantic tags: or use `role` attribute
- Use labels
- Use alt for images
- `area-hidden`, `area-label`, ... attributes for dynamic content

# Accessibility - Compliance check
















# Assignment 8 solution walkthrough

🔗 <info/examples/bigexample>

- Provide login and backend for contact book

Logged inn as: johndoe Logout

 Name ▾  

<b>Don John</b> 12-322-622 <a href="mailto:don.john@ymail.com">don.john@ymail.com</a>	 
<b>Elizabeth Westland</b> 66-112-312 <a href="mailto:e47wl@outlook.com">e47wl@outlook.com</a>	 
<b>John Smith</b> 12-345-678 <a href="mailto:john.smith@gmail.com">john.smith@gmail.com</a>	 
<b>Kevin Magnussen</b> <a href="mailto:kevinrulez@noemail.com">kevinrulez@noemail.com</a> <a href="tel:+319971121">+31 997-11-21</a>	 

# Assignment 8 solution walkthrough

🔗 [assignments/assignment-8/8-solution](#)

- Database contains

- Users

```
CREATE TABLE users (  
    userid INTEGER,  
    username TEXT NOT NULL,  
    passwordhash TEXT NOT NULL,  
    PRIMARY KEY(userid)  
    UNIQUE(username)  
)
```

- Addresses

```
CREATE TABLE addresses (  
    addressid INTEGER,  
    user INTEGER,  
    name TEXT NOT NULL,  
    tel TEXT,  
    email TEXT,  
    PRIMARY KEY(addressid)  
    FOREIGN KEY (user) REFERENCES users (userid)  
)
```

# Assignment 8 solution walkthrough

🔗 [assignments/assignment-8/8-solution](#)

## - **app.py** routes

```
# The first three routes do not fit into a rest API.
```

```
@app.route("/")  
def index():
```

```
@app.route("/login", methods=["POST"])  
def login():
```

/login returns user data from db.

```
@app.route("/logout")  
def logout():
```

# Assignment 8 solution walkthrough

🔗 assignments/assignment-8/8-solution

## - app.py routes

```
# Here comes the rest API for addresses.  
@app.route("/user", methods=["GET"])  
def get_user():
```

/user gets logged in userid from session and returns user data from db.

```
@app.route("/addresses", methods=["GET"])  
def get_addresses():
```

/addresses [GET] returns all addresses for logged in userid

```
@app.route("/addresses", methods=["POST"])  
def add_address():
```

/addresses [POST] adds new address and returns new id

```
@app.route("/addresses/<addressid>", methods=["PUT"])  
def set_address(addressid):
```

/addresses/<id> [PUT] updates address

```
@app.route("/addresses/<addressid>", methods=["DELETE"])  
def del_address(addressid):
```

# Assignment 8 solution walkthrough

🔗 [assignments/assignment-8/8-solution](#)

- **data.js** ajax calls for each endpoint

```
async function login(){ ... }

async function logout(){ ... }

// try if there is a user logged in
async function getUser(){... }

async function getAddresses(){... }

async function storeAddress(address){ ... }

async function updateAddress(address){ ... }

async function deleteAddress(address){ ... }
```

# Assignment 8 solution walkthrough

🔗 [assignments/assignment-8/8-solution](#)

- **data.js** ajax calls for each endpoint
- **catch** network error
- **alert** on error
- **return true** or **false** to signal success

```
async function deleteAddress(address){
  try {
    let url = "/addresses/" + address.id;
    let response = await fetch(url, {
      method: "DELETE",
    });
    if (response.status == 200){
      return true;
    }
    console.log("Error deleting contact.")
    alert("Error deleting contact, please refresh
page.")
  } catch (error) {
    console.log(error);
    alert(`Network error: ${error}`)
  }
  return false;
}
```

# Assignment 8 solution walkthrough

🔗 assignments/assignment-8/8-solution

## Login flow:

- Check if a user is already logged in:

```
<body onload="getUser()">
```

- Allow for manual log in:

```
<form action="javascript:void(0);" onsubmit="login();">
```

- After login, hide login form - with userid get addresses:

```
async function login(){
  form.style.display = "none";

  try {
    ...
    getAddresses();
  } catch (error) { ... }
}
```

# Assignment 8 solution walkthrough

🔗 [assignments/assignment-8/8-solution](#)

## Flaws

- Solution does not show a loading icon, while AJAX requests are ongoing.
- Error messages could be better displayed, not as Alert, but on page.