

Introduction to Internet Applications

Internet Applications, ID1354

Contents

- Distributed Architectures
- User Interface Design
- Tools

Distributed
Architectures

User Interface
Design

Tools

Section

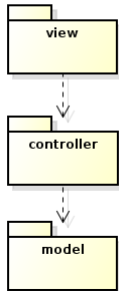
- **Distributed Architectures**
- User Interface Design
- Tools

Distributed
Architectures

User Interface
Design

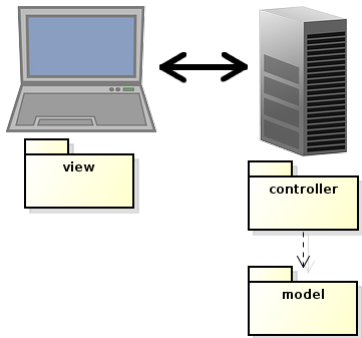
Tools

Local Application



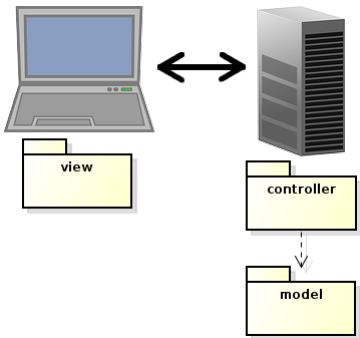
- ▶ We are familiar with an architecture where the entire application resides **on the same computer**.

Introducing a Server



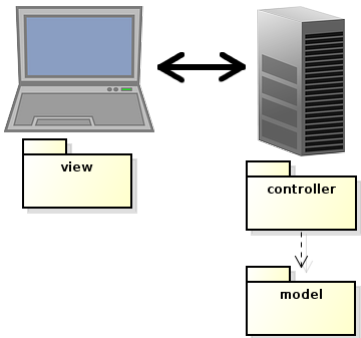
- ▶ Now, the application will be **split on two tiers** (computers).

Introducing a Server



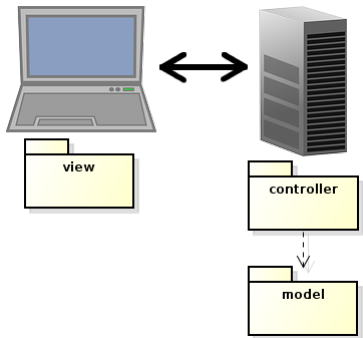
- ▶ Now, the application will be **split on two tiers** (computers).
- ▶ A **client** that has the view and a **server** that has controller and model.

Introducing a Server



- ▶ Now, the application will be **split on two tiers** (computers).
- ▶ A **client** that has the view and a **server** that has controller and model.
- ▶ The view is displayed in a **web browser**.

Introducing a Server



- ▶ Now, the application will be **split on two tiers** (computers).
- ▶ A **client** that has the view and a **server** that has controller and model.
- ▶ The view is displayed in a **web browser**.

This architecture is not good, we also need layers for communication.

Server-Side Communication



- ▶ First, we add a server layer, normally called **view** (a bit confusing).

Server-Side Communication



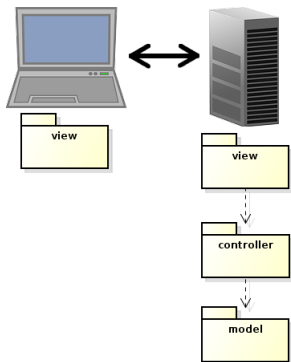
- ▶ First, we add a server layer, normally called **view** (a bit confusing).
- ▶ However, the server side view layer performs tasks typical of a view:

Server-Side Communication



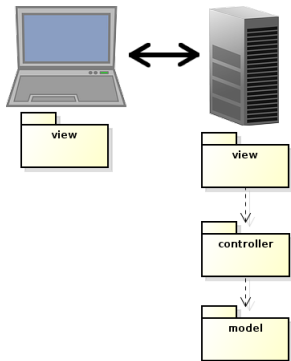
- ▶ First, we add a server layer, normally called **view** (a bit confusing).
- ▶ However, the server side view layer performs tasks typical of a view:
 - ▶ **Creates views** (HTML), which are sent to the client.

Server-Side Communication



- ▶ First, we add a server layer, normally called **view** (a bit confusing).
- ▶ However, the server side view layer performs tasks typical of a view:
 - ▶ **Creates views** (HTML), which are sent to the client.
 - ▶ **Interprets user gestures**, a click in a web page creates a request to the server.

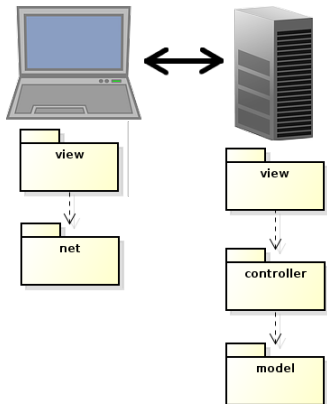
Server-Side Communication



- ▶ First, we add a server layer, normally called **view** (a bit confusing).
- ▶ However, the server side view layer performs tasks typical of a view:
 - ▶ **Creates views** (HTML), which are sent to the client.
 - ▶ **Interprets user gestures**, a click in a web page creates a request to the server.

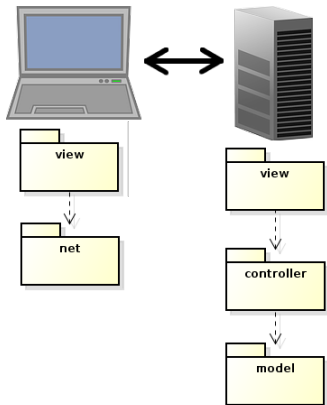
It might seem that we need yet a layer, for network handling. There is such a layer, but it is in the web server. We don't write it ourselves.

Client-Side Communication



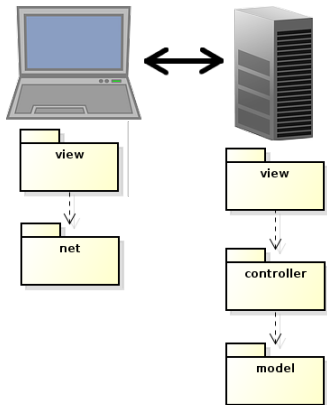
- ▶ Next, we add a client layer for communication, the **net** layer.

Client-Side Communication



- ▶ Next, we add a client layer for communication, the **net** layer.
- ▶ Actually, the browser handles most of the communication.
 - ▶ The small network code written by us is normally considered part of the client-side view, the **net layer is omitted**.

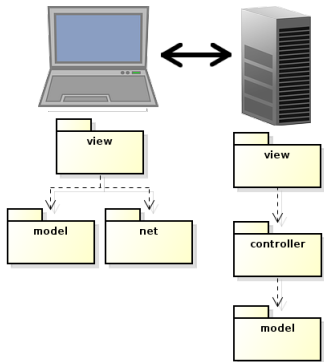
Client-Side Communication



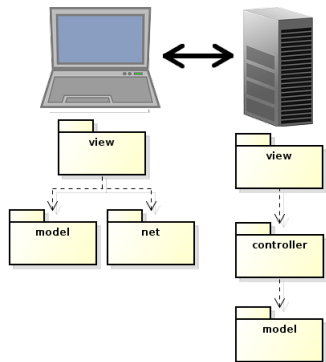
- ▶ Next, we add a client layer for communication, the **net** layer.
- ▶ Actually, the browser handles most of the communication.
 - ▶ The small network code written by us is normally considered part of the client-side view, the **net layer is omitted**.
- ▶ This is a traditional web application.

The MVVM Pattern

- ▶ The trend is that data is stored also on the client, therefore we get a **client-side model**.

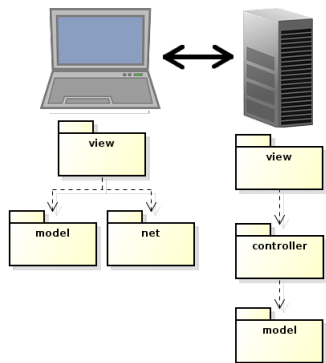


The MVVM Pattern



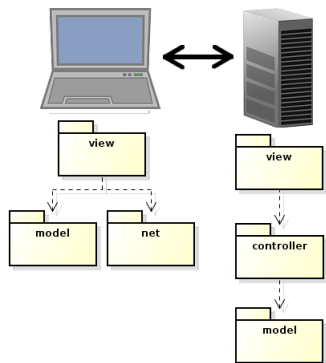
- ▶ The trend is that data is stored also on the client, therefore we get a **client-side model**.
- ▶ This reduces the network communication, since we do **not need to resend the entire view** each time the user does something.

The MVVM Pattern



- ▶ The trend is that data is stored also on the client, therefore we get a **client-side model**.
- ▶ This reduces the network communication, since we do **not need to resend the entire view** each time the user does something.
- ▶ Thereby, the application becomes faster.

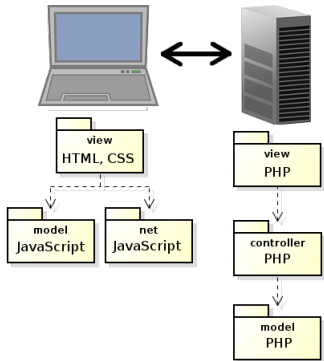
The MVVM Pattern



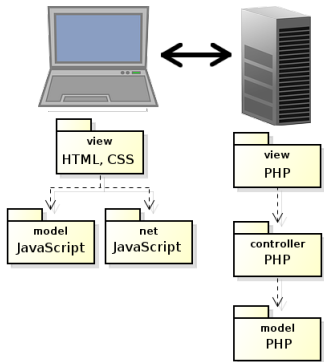
- ▶ The trend is that data is stored also on the client, therefore we get a **client-side model**.
- ▶ This reduces the network communication, since we do **not need to resend the entire view** each time the user does something.
- ▶ Thereby, the application becomes faster.
- ▶ This is referred to as the **MVVM**, model-view-viewmodel pattern.

Programming Languages

- ▶ This is the architecture we will normally use during the course.

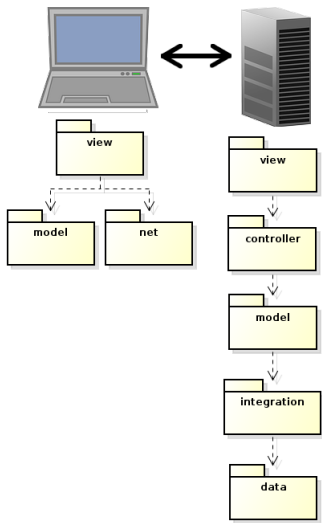


Programming Languages



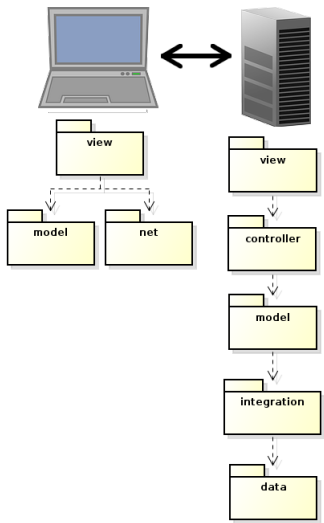
- ▶ This is the architecture we will normally use during the course.
- ▶ The view is programmed in **HTML** and **CSS**, client side behavior is programmed in **JavaScript** and the entire server side code is written in **PHP**.

Three-Tier Architecture



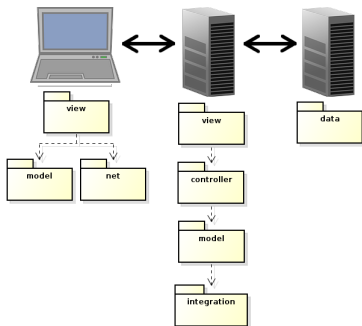
- ▶ Of course, we also need to store data. That is done in the **data** layer, which is often a database.

Three-Tier Architecture



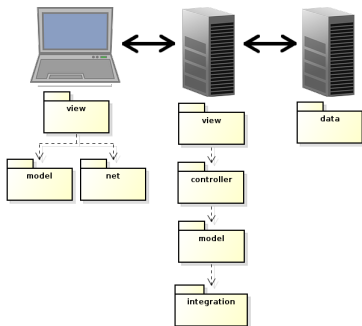
- ▶ Of course, we also need to store data. That is done in the **data** layer, which is often a database.
- ▶ We also introduce the **integration** layer, to handle the database calls.

Three-Tier Architecture (Cont'd)



- ▶ In a bigger application, we would most likely place the database in a separate node.

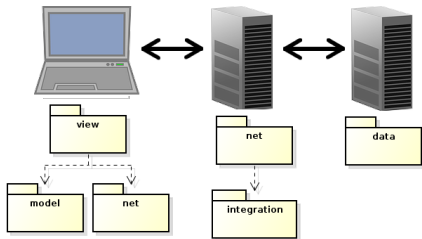
Three-Tier Architecture (Cont'd)



- ▶ In a bigger application, we would most likely place the database in a separate node.
- ▶ This is called **three-tier architecture** and is, since long time, the **dominating architecture** for web applications.

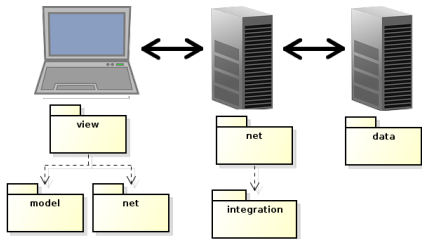
Question 1

Event-Driven Architecture



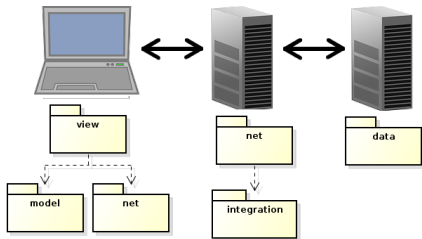
- ▶ There is a growing tendency to move business logic to the client, maybe even completely remove the server-side model.

Event-Driven Architecture



- ▶ There is a growing tendency to move business logic to the client, maybe even completely remove the server-side model.
- ▶ This is made possible with **web sockets**, which enable **full duplex** browser-server communication.

Event-Driven Architecture



- ▶ There is a growing tendency to move business logic to the client, maybe even completely remove the server-side model.
- ▶ This is made possible with [web sockets](#), which enable [full duplex](#) browser-server communication.
- ▶ The motive is to reduce communication latency. The browser informs the server about user actions, but does [not wait for response](#) before updating the view.

Section

- Distributed Architectures
- **User Interface Design**
- Tools

Use UI Guidelines!

- ▶ This is not a course in human-computer interaction. Still, it is mandatory to consider basic **heuristics for user interface design**.

Use UI Guidelines!

- ▶ This is not a course in human-computer interaction. Still, it is mandatory to consider basic [heuristics for user interface design](#).
- ▶ There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:

Use UI Guidelines!

- ▶ This is not a course in human-computer interaction. Still, it is mandatory to consider basic **heuristics for user interface design**.
- ▶ There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
 - ▶ **10 Usability Heuristics for UI Design**,
`http://www.nngroup.com/articles/ten-usability-heuristics/`

Use UI Guidelines!

- ▶ This is not a course in human-computer interaction. Still, it is mandatory to consider basic **heuristics for user interface design**.
- ▶ There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
 - ▶ **10 Usability Heuristics for UI Design**,
`http://www.nngroup.com/articles/ten-usability-heuristics/`
 - ▶ **Top 10 Guidelines for Homepage Usability**,
`http://www.nngroup.com/articles/top-ten-guidelines-for-homepage-usability/`

Use UI Guidelines!

- ▶ This is not a course in human-computer interaction. Still, it is mandatory to consider basic **heuristics for user interface design**.
- ▶ There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
 - ▶ **10 Usability Heuristics for UI Design**,
`http://www.nngroup.com/articles/ten-usability-heuristics/`
 - ▶ **Top 10 Guidelines for Homepage Usability**,
`http://www.nngroup.com/articles/top-ten-guidelines-for-homepage-usability/`
 - ▶ **Top 10 Mistakes in Web Design**,
`http://www.nngroup.com/articles/top-10-mistakes-web-design/`

1. Visibility of system status

J. Nielsen's 10 Usability Heuristics

- ▶ The system should always **keep users informed** about what is going on, through appropriate feedback.

1. Visibility of system status

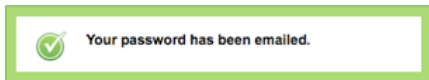
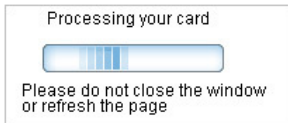
J. Nielsen's 10 Usability Heuristics

- ▶ The system should always **keep users informed** about what is going on, through appropriate feedback.
- ▶ The **UI must change** within one second after a user action, or the user might think nothing happened.

1. Visibility of system status

J. Nielsen's 10 Usability Heuristics

- ▶ The system should always **keep users informed** about what is going on, through appropriate feedback.
- ▶ The **UI must change** within one second after a user action, or the user might think nothing happened.
- ▶ **Good examples:**



2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- ▶ Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- ▶ Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- ▶ Good example (“How can we help you?” better than “FAQ”):

How can we help you?

2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- ▶ Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- ▶ Good example (“How can we help you?” better than “FAQ”):
- ▶ Bad example (“Continue if enabled” is system oriented language):
oops, there is a problem

Target.com requires **cookie** to be enabled.

continue if enabled

3. User control and freedom

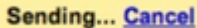
J. Nielsen's 10 Usability Heuristics

- ▶ We often do things by mistake, and therefore need a **clearly marked “emergency exit”** to leave an unwanted state without having to go through an extended dialogue. Support undo and redo.

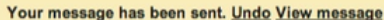
3. User control and freedom

J. Nielsen's 10 Usability Heuristics

- ▶ We often do things by mistake, and therefore need a **clearly marked “emergency exit”** to leave an unwanted state without having to go through an extended dialogue. Support undo and redo.
- ▶ Good examples:



Sending... [Cancel](#)



Your message has been sent. [Undo](#) [View message](#)

4. Consistency and standards

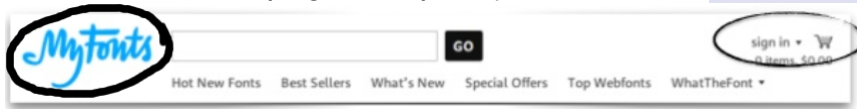
J. Nielsen's 10 Usability Heuristics

- ▶ Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

4. Consistency and standards

J. Nielsen's 10 Usability Heuristics

- ▶ Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- ▶ Good example (Sign in at top right, logo with link to index page at top left):



5. Error prevention

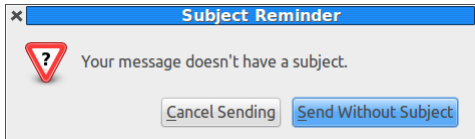
J. Nielsen's 10 Usability Heuristics

- ▶ Create a careful design which **prevents problems from occurring**. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

5. Error prevention

J. Nielsen's 10 Usability Heuristics

- ▶ Create a careful design which **prevents problems from occurring**. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- ▶ **Good example:**



6. Recognition rather than recall

J. Nielsen's 10 Usability Heuristics

- ▶ Minimize the user's memory load by **making objects, actions, and options visible**. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

6. Recognition rather than recall

J. Nielsen's 10 Usability Heuristics

- ▶ Minimize the user's memory load by **making objects, actions, and options visible**. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- ▶ **Good example:**

The screenshot shows a hotel search form with the following elements:

- Hotel search** (Section Header)
- Destination input field with placeholder text: "Destination, hotel, landmark or address"
- Check in** (Section Header) with date input: "13/08/2015" and day: "Thursday"
- Check out** (Section Header) with date input: "14/08/2015" and day: "Friday"
- Night** (Section Header) with a calendar icon and a "1" icon
- Rooms** (Section Header) with a dropdown menu: "1 room, 2 adults"
- Search** (Button)

- ▶ Clear headline.
- ▶ No doubt where to click to start the search.

7. Flexibility and efficiency of use

J. Nielsen's 10 Usability Heuristics

- ▶ **Accelerators**, unseen by the novice user, may often **speed up the interaction for the expert** user.

7. Flexibility and efficiency of use

J. Nielsen's 10 Usability Heuristics

- ▶ **Accelerators**, unseen by the novice user, may often **speed up the interaction for the expert** user.
- ▶ Allow the user to **change** the accelerators.

7. Flexibility and efficiency of use

J. Nielsen's 10 Usability Heuristics

- ▶ **Accelerators**, unseen by the novice user, may often **speed up the interaction for the expert** user.
- ▶ Allow the user to **change** the accelerators.
- ▶ **Examples are**
 - ▶ *Saved searches*
 - ▶ *Items you recently looked at*
 - ▶ *Save query for later*

8. Aesthetic and minimalist design

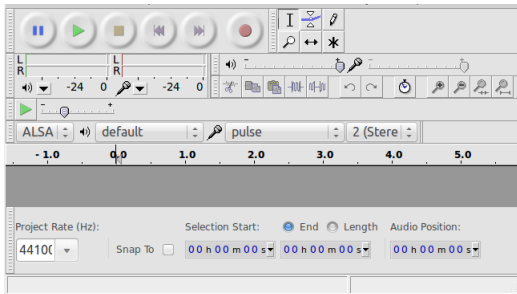
J. Nielsen's 10 Usability Heuristics

- ▶ Dialogues should **not contain information which is irrelevant or rarely needed**. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their visibility.

8. Aesthetic and minimalist design

J. Nielsen's 10 Usability Heuristics

- ▶ Dialogues should **not contain information which is irrelevant or rarely needed**. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their visibility.
- ▶ **Bad example:**



9. Help users recognize, diagnose, and recover from errors

J. Nielsen's 10 Usability Heuristics

- ▶ **Error messages** should be expressed in plain language (**no codes**), precisely indicate the problem, and constructively suggest a solution.

9. Help users recognize, diagnose, and recover from errors

J. Nielsen's 10 Usability Heuristics

- ▶ **Error messages** should be expressed in plain language (**no codes**), precisely indicate the problem, and constructively suggest a solution.
- ▶ **Do not** tell the user *unexpected exception* or anything similar.

9. Help users recognize, diagnose, and recover from errors

J. Nielsen's 10 Usability Heuristics

- ▶ **Error messages** should be expressed in plain language (**no codes**), precisely indicate the problem, and constructively suggest a solution.
- ▶ **Do not** tell the user *unexpected exception* or anything similar.
- ▶ **Good examples:**

Please ensure all fields highlighted in red are filled.

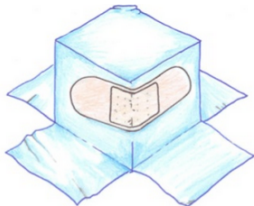
Email: *

Telephone:

D.O.B: DD ▼ MM ▼ YYYY ▼

Select the nature of your request:

Further Education Research request



Error

Something went wrong. Don't worry, your files are still safe and the Dropboxers have been notified. Check out our [Help Center](#) and [forums](#) for help, or head back to [home](#).

10. Help and documentation

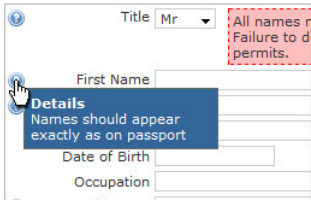
J. Nielsen's 10 Usability Heuristics

- ▶ Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

10. Help and documentation

J. Nielsen's 10 Usability Heuristics

- ▶ Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
- ▶ Good example:



The screenshot shows a form with several input fields: Title (Mr), First Name, Date of Birth, and Occupation. A tooltip is displayed over the First Name field, stating "Details: Names should appear exactly as on passport". A red dashed box highlights an error message: "All names r Failure to d permits.".

Question 2

Section

- Distributed Architectures
- User Interface Design
- **Tools**

Web Development Tools

- ▶ There are many tools that facilitates developing web applications.

Web Development Tools

- ▶ There are many tools that facilitates developing web applications.
- ▶ You are strongly advised to start using some of the following tools, they will help you a lot.

Web Development Tools

- ▶ There are many tools that facilitates developing web applications.
- ▶ You are strongly advised to start using some of the following tools, they will help you a lot.
- ▶ **Firefox is used at lectures, but there are similar tools in other browsers as well.**

Development Tools or Browser Web Console

Distributed
ArchitecturesUser Interface
Design

Tools

- ▶ Lets you **choose elements** from the web page and have their HTML and CSS displayed.

The screenshot shows the browser's developer tools interface. The top pane displays the HTML structure of the page, with the following code visible:

```

<body class="DefaultTheme startDepartment lang-sv-SE use-personal-menu">
  <div id="fb-root" class="fb-root"></div>
  <div id="page" class="content"></div>
  <div class="logImage"></div>
  <div id="backToTop" href="#" title="Till sidans topp">Till sidans topp</div>
</body>
  
```

The middle pane shows the CSS for the selected element, including the Box Model diagram. The diagram illustrates the element's dimensions and spacing:

- margin: 29.7px
- padding: 0px
- border: 1px solid black
- width: 883px
- height: 2418.05px

The bottom pane shows the console with the following error messages:

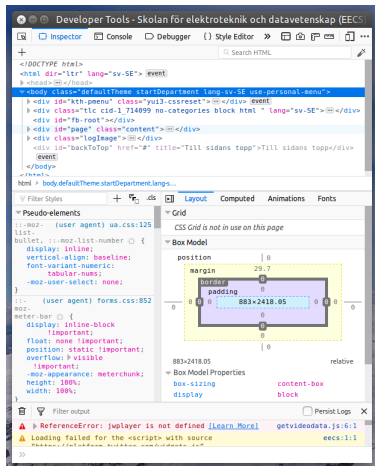
```

ReferenceError: jquery is not defined | Learn More | getvideodata.js:6:1
Loading failed for the <script> with source | eecs:1:1
  
```

Development Tools or Browser Web Console

Distributed
ArchitecturesUser Interface
Design

Tools

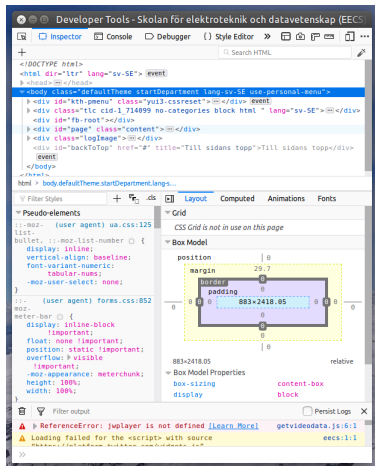


- ▶ Lets you **choose elements** from the web page and have their HTML and CSS displayed.
- ▶ Possible to **update HTML and CSS**.

Development Tools or Browser Web Console

Distributed
ArchitecturesUser Interface
Design

Tools



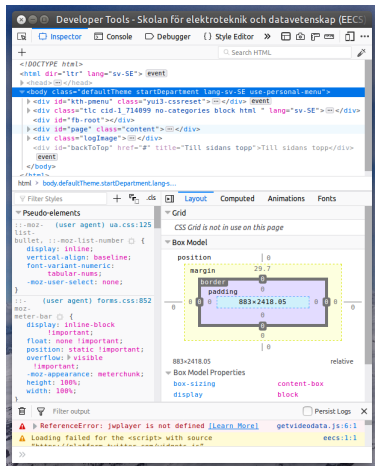
- ▶ Lets you choose elements from the web page and have their HTML and CSS displayed.
- ▶ Possible to update HTML and CSS.
- ▶ Logs all Http requests and responses.

Development Tools or Browser Web Console

Distributed Architectures

User Interface Design

Tools



- ▶ Lets you choose elements from the web page and have their HTML and CSS displayed.
- ▶ Possible to update HTML and CSS.
- ▶ Logs all Http requests and responses.
- ▶ Includes a JavaScript debugger.

Development Tools or Browser Web Console

Distributed Architectures

User Interface Design

Tools

The screenshot shows the browser's developer tools interface. The top panel displays the HTML structure of the page, with the following code visible:

```

<body class="DefaultTheme.startDepartment lang-sv-SE use-personal-menu">
  <div id="fb-root"></div>
  <div id="page" class="content"></div>
  <div class="logImage"></div>
  <div id="backToTop" href="#" title="Till sidans topp">Till sidans topp</div>
</body>

```

The middle panel shows the CSS styles for the selected element, including:

```

display: inline-block;
float: none;
position: static;
overflow: visible;
width: 100%;
height: 100%;

```

The right panel shows the Box Model diagram for the selected element, with the following dimensions:

- margin: 0
- padding: 0
- border: 0
- width: 883px
- height: 2418.05px

The bottom panel shows the console log with the following error:

```

ReferenceError: jquery is not defined
Loading failed for the <script> with source

```

▶ Lets you choose elements from the web page and have their HTML and CSS displayed.

▶ Possible to update HTML and CSS.

▶ Logs all Http requests and responses.

▶ Includes a JavaScript debugger.

▶ Possible to add and delete cookies.

Development Tools (Cont'd)

- ▶ The console is opened with **Ctrl-Shift-K** in Firefox and **Ctrl-Shift-J** in Chrome.

Validators

- ▶ There are [online validators](#) for both HTML and CSS. Links can be found on the course web site.

Validators

- ▶ There are [online validators](#) for both HTML and CSS. Links can be found on the course web site.
- ▶ Remember to [always validate](#) your HTML and CSS code.

PhpStorm

- ▶ There are many different IDEs for web development, all have their pros and cons.

PhpStorm

- ▶ There are many different IDEs for web development, all have their pros and cons.
- ▶ PhpStorm will be used for examples during the course.

PhpStorm

- ▶ There are many different IDEs for web development, all have their pros and cons.
- ▶ PhpStorm will be used for examples during the course.
- ▶ Most important is that you actually use an IDE, do not program in a text editor unless you are really sure it is what you prefer.

JSFiddle and JSLint

- ▶ **JSFiddle** is an **online editor** where you can test HTML, CSS and JavaScript.

JSFiddle and JSLint

- ▶ **JSFiddle** is an **online editor** where you can test HTML, CSS and JavaScript.
- ▶ **JSLint** is an online tool for testing JavaScript **code quality**.

W3Schools Try It Yourself

- ▶ **w3schools.com** has **excellent tutorials** for all languages covered in the course.

W3Schools Try It Yourself

- ▶ `w3schools.com` has [excellent tutorials](#) for all languages covered in the course.
- ▶ All examples are presented with an [online editor](#) where you can experiment with your code.