JDOM

Motivation

- We need a programming model for manipulating XML
- We want it to be:
  - Simple
  - Object-oriented and based on our understanding of XML
- We should be able to:
  - Change and store the changed XML
  - Reuse our knowledge to program with XML in other languages

The JDOM Framework

- An implementation of **generic XML trees** in Java
- Nodes are represented as **classes** and ***interfaces***
- Change XML by changing Java objects
  - Store back to disk or send to client afterwards

- DOM is a *language-independent* alternative
  - We will look at DOM when we study JavaScript

Overview

- **Classes and interfaces**
- Selecting nodes
- Reading and storing XML
- XML Schema validation
- Example application

JDOM Classes and Interfaces

- The abstract class **`Content`** has subclasses:
  - `Comment`
  - `DocType`
  - **`Element`**
  - `EntityRef`
  - `ProcessingInstruction`
  - **`Text`**
- Other classes are `Attribute` and `Document`
- The `Parent` interface describes `Document` and `Element`
- `All in the` **`org.jdom2`** `package`

Straight-forward API

- To create a *luke* element:

```
Element luke = new Element("luke");
```

- To create a *vader* element and set *luke* as child:

```
Element vader = new Element("vader");
vader.addContent(luke);
```

- That is:

```
<vader>
        <luke />
</vader>
```

Including namespaces

- To refer to a namespace:

```
String uri = "http://i.am";
Namespace ns = Namespace.getNamespace(uri);
```

- To create a *groot* element in this namespace:

```
Element groot = new Element("groot",ns);
groot.addContent(new Element("groot",ns);
```

```
<groot xmlns="http://i.am">
       <groot />
</groot >
```

Attributes

- Use getAttribute/setAttribute on Element:

```
Element foo = new Element("foo");
foo.setAttribute("bar","baz");
```

- Result:

```
<foo bar="baz" />
```

- Get the value:

```
Attribute a = foo.getAttribute("bar");
String value = a.getValue(); // "baz"
```

- There are also a namespace-aware versions

A Simple Example

```
int xmlHeight(Element e) {
  List<Content> contents = e.getContent();
  int max = 0;
  for (Content c : contents) {
    int h;
    if (c instanceof Element)
      h = xmlHeight((Element)c);
    else
      h = 1;
    if (h > max)
      max = h;
  }
  return max+1;
}
```

Another Example (1/3)

- Modify all elements like

```
<ingredient name="butter" amount="0.25" unit="cup"/>
```

into a more elaborate version:

```
<ingredient name="butter">
  <ingredient name="cream" unit="cup" amount="0.5" />
  <preparation>
    Churn until the cream turns to butter.
  </preparation>
</ingredient>
```

Another Example (2/3)

```
void makeButter(Element e) throws DataConversionException {
  Namespace rcp =
    Namespace.getNamespace("http://www.brics.dk/ixwt/recipes");
  ListIterator<Element> i = e.getChildren().listIterator();
  while (i.hasNext()) {
    Element c = i.next();
    if (c.getName().equals("ingredient") &&
        c.getAttributeValue("name").equals("butter")) {
      Element butter = new Element("ingredient",rcp);
      butter.setAttribute("name","butter");
```

Another Example (3/3)

```
      Element cream = new Element("ingredient",rcp);
      cream.setAttribute("name","cream");
      cream.setAttribute("unit",c.getAttributeValue("unit"));
      double amount = c.getAttribute("amount").getDoubleValue();
      cream.setAttribute("amount",new Double(2*amount).toString());
      butter.addContent(cream);
      Element churn = new Element("preparation",rcp);
      churn.addContent("Churn until the cream turns to butter.");
      butter.addContent(churn);
      i.set((Element)butter);
    } else {
      makeButter(c);
    }
  }
}
```

Clicker question

- Which property holds for all programs that use JDOM to generate XML?
  - They always terminate and return an XML document.
  - They always generate well-formed XML. ✓
  - They always generate valid XML.

Overview

- Classes and interfaces
- **Selecting nodes**
- Reading and storing XML
- XML Schema validation
- Example application

Selecting nodes in JDOM

Not part of the exam

- Languages for node selection
  - CSS selectors
    - For CSS stylesheets
    - Later on the course: used for node selection in Javascript with jQuery
  - XPath
    - Generic selection language for XML languages
    - (Much) more powerful than CSS selectors
    - Used in JDOM (and many other places)

- We will only cover a small (but useful) subset of Xpath
  - Read more in IXWT chapter 3

XPath crash course (1/2)

Not part of the exam

- In XPath 1.0 namespace prefixes must be used!
  - Assume here that h is bound the XHTML namespace
- Select all (XHTML) a elements

XPath: `//h:a`	CSS: `a`

- Select all a that are descendants of b

XPath: `//h:b//h:a`	CSS: `b a`

- Select all a that are children of b

XPath: `//h:b/h:a`	CSS: `b > a`

- Select all a that are parents of b

XPath: `//h:b/parent::h:a`	CSS: `Not possible`

XPath crash course (2/2)

Not part of the exam

- Select all a elements that have a foo attribute

XPath: `//h:a[@foo]`	CSS: `a[foo]`

- Select all foo attributes of a elements

XPath: `//h:a/@foo`	CSS: `Not possible`

- Can combine selectors:

XPath: `//h:b/h:a[@foo]`	CSS: `b > a[foo]`

- XPath has concept of context. Select b in context:

XPath: `.//h:b`	CSS: `N/A`

- A context can be an element somewhere in the tree
- We will see a similar context with CSS in jQuery

XPath Evaluation

Not part of the exam

```
void removeSugar(Document d) throws JDOMException {
  XPathFactory fac = XPathFactory.instance();
  Namespace ns = Namespace
         .getNamespace("rcp","http://www.brics.dk/ixwt/recipes");
  XPathExpression<Element> exp = fac
                 .compile("//rcp:ingredient[@name='sugar']",
                          null,
                          Filters.element(),
                          ns);
  for (Element e : exp.evaluate(d)) {
    double amount = e.removeAttribute("amount");
  }
}
```

Only return elements (not e.g. attributes)

Whole d is context

Needs Xpath processsor (Jaxen) in classpath

Overview

- Classes and interfaces
- Selecting nodes
- **Reading and storing XML**
- XML Schema validation
- Example application

Parsing and Serializing

```
public class ChangeDescription {
  public static void main(String[] args) {
    try {
      SAXBuilder b = new SAXBuilder();
      Document d = b.build(new File("recipes.xml"));
      Namespace rcp =
        Namespace.getNamespace("http://www.brics.dk/ixwt/recipes");
      d.getRootElement().getChild("description",rcp)
                        .setText("Cool recipes!");
      XMLOutputter outputter = new XMLOutputter();
      outputter.output(d, System.out);
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

Overview

- Classes and interfaces
- Selecting nodes
- Reading and storing XML
- **XML Schema validation**
- Example application

Validation (XML Schema)

```
public class ValidateXMLSchema {
  public static void main(String[] args) {
    try {
      SAXBuilder b = new SAXBuilder();
      b.setValidation(true);
      b.setProperty(
        "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
        "http://www.w3.org/2001/XMLSchema");
      b.setProperty(
        "http://java.sun.com/xml/jaxp/properties/schemaSource",
        new File(args[1]));
      String msg = "No errors!";
      try {
        Document d = b.build(new File(args[0]));
      } catch (JDOMParseException e ) {
        msg = e.getMessage();
      }
      System.out.println(msg);
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

Validation (with JDOM classes)

```
public class ValidateXMLSchema {
  public static void main(String[] args) {
    try {
      File xsdfile = new File(args[1]);
      XMLReaderJDOMFactory schemafac = new XMLReaderXSDFactory(xsdfile);
      SAXBuilder builder = new SAXBuilder(schemafac);

      String msg = "No errors!";
      try {
        Document d = b.build(new File(args[0]));
      } catch (JDOMParseException e ) {
        msg = e.getMessage();
      }
      System.out.println(msg);
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

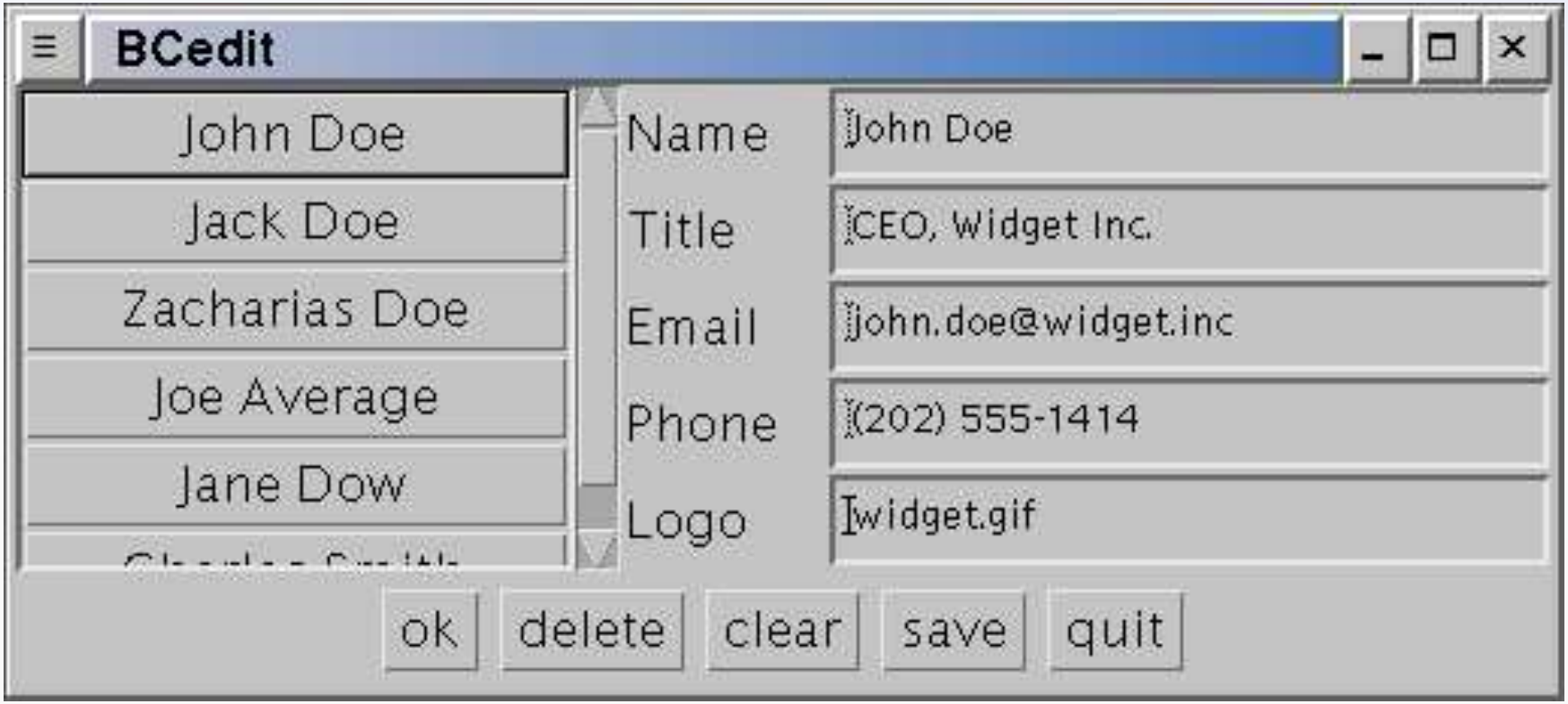
Overview

- Classes and interfaces
- Selecting nodes
- Reading and storing XML
- XML Schema validation
- **Example application**

Business Cards

```
<cardlist xmlns="http://businesscard.org"
          xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <title>
    <xhtml:h1>My Collection of Business Cards</xhtml:h1>
    containing people from <xhtml:em>Widget Inc.</xhtml:em>
  </title>
  <card>
    <name>John Doe</name>
    <title>CEO, Widget Inc.</title>
    <email>john.doe@widget.com</email>
    <phone>(202) 555-1414</phone>
  </card>
  <card>
    <name>Joe Smith</name>
    <title>Assistant</title>
    <email>thrall@widget.com</email>
  </card>
</cardlist>
```

Business Card Editor



Class Representation

```
class Card {
  public String name,title,email,phone,logo;

  public Card(String name, String title, String email,
              String phone, String logo) {
    this.name=name;
    this.title=title;
    this.email=email;
    this.phone=phone;
    this.logo=logo;
  }
}
```

From JDOM to Classes

```
Vector doc2vector(Document d) {
   Vector v = new Vector();
   Iterator i = d.getRootElement().getChildren().iterator();
   while (i.hasNext()) {
     Element e = (Element)i.next();
     String phone = e.getChildText("phone",b);
     if (phone==null) phone="";
     Element logo = e.getChild("logo",b);
     String uri;
     if (logo==null) uri="";
     else uri=logo.getAttributeValue("uri");
     Card c = new Card(e.getChildText("name",b),
                       e.getChildText("title",b),
                       e.getChildText("email",b),
                       phone, uri);
     v.add(c);
   }
   return v;
 }
```

From Classes to JDOM (1/2)

```
Document vector2doc() {
   Element cardlist = new Element("cardlist");
   for (int i=0; i<cardvector.size(); i++) {
     Card c = (Card)cardvector.elementAt(i);
     if (c!=null) {
       Element card = new Element("card",b);
       Element name = new Element("name",b);
       name.addContent(c.name); card.addContent(name);
       Element title = new Element("title",b);
       title.addContent(c.title); card.addContent(title);
       Element email = new Element("email",b);
       email.addContent(c.email); card.addContent(email);
```

From Classes to JDOM (2/2)

```
      if (!c.phone.equals("")) {
         Element phone = new Element("phone",b);
         phone.addContent(c.phone);
         card.addContent(phone);
       }
       if (!c.logo.equals("")) {
         Element logo = new Element("logo",b);
         logo.setAttribute("uri",c.logo);
         card.addContent(logo);
       }
       cardlist.addContent(card);
     }
   }
   return new Document(cardlist);
}
```

A Little Bit of Code

```
void addCards() {
   cardpanel.removeAll();
   for (int i=0; i<cardvector.size(); i++) {
     Card c = (Card)cardvector.elementAt(i);
     if (c!=null) {
       Button b = new Button(c.name);
       b.setActionCommand(String.valueOf(i));
       b.addActionListener(this);
       cardpanel.add(b);
     }
   }
   this.pack();
 }
```

The Main Application

```
public BCedit(String cardfile) {
   super("BCedit");
   this.cardfile=cardfile;
   try {
     cardvector = doc2vector(
        new SAXBuilder().build(new File(cardfile)));
   } catch (Exception e) { e.printStackTrace(); }
   // initialize the user interface
   ...
 }
```

Summary

- Classes and interfaces
- Selecting nodes
- Reading and storing XML
- XML Schema validation
- Example application

DOM programming on other languages

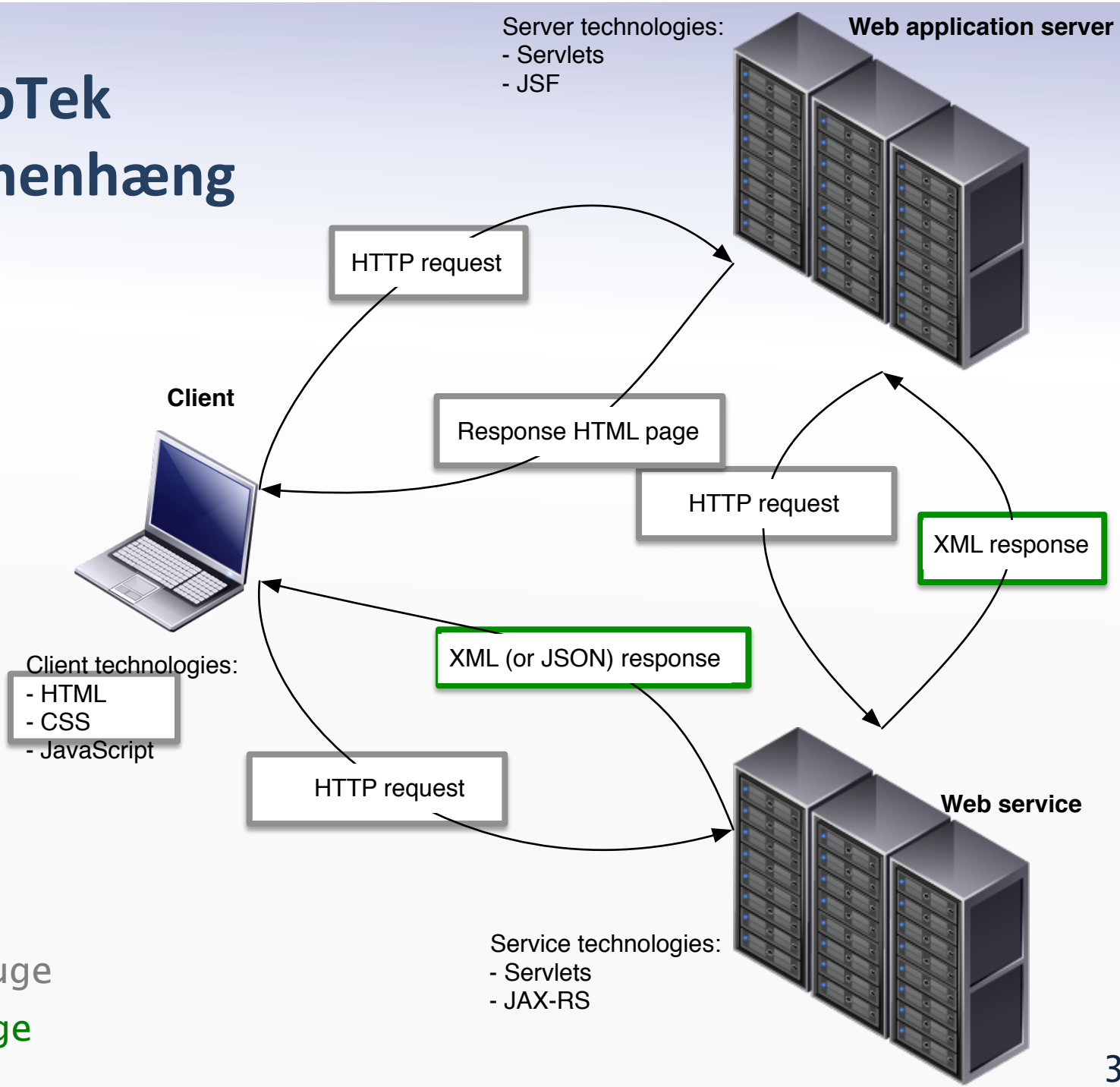
- JDOM is a Java framework, **but**:
  - Nothing inherently Java-bound in the model
  - DOM programming exists in all modern languages
    - (Very) similar API, different language...

- Other languages with DOM frameworks:
  - C#
  - PHP
  - Python
  - ...
  - **JavaScript** (we will see this later in the course)

Essential Online Resources

- JDOM: http://jdom.org/
- XPath processors:
  - XPath 1.0 http://jaxen.codehaus.org/
  - XPath 2.0 and 3.0: http://saxonica.com/

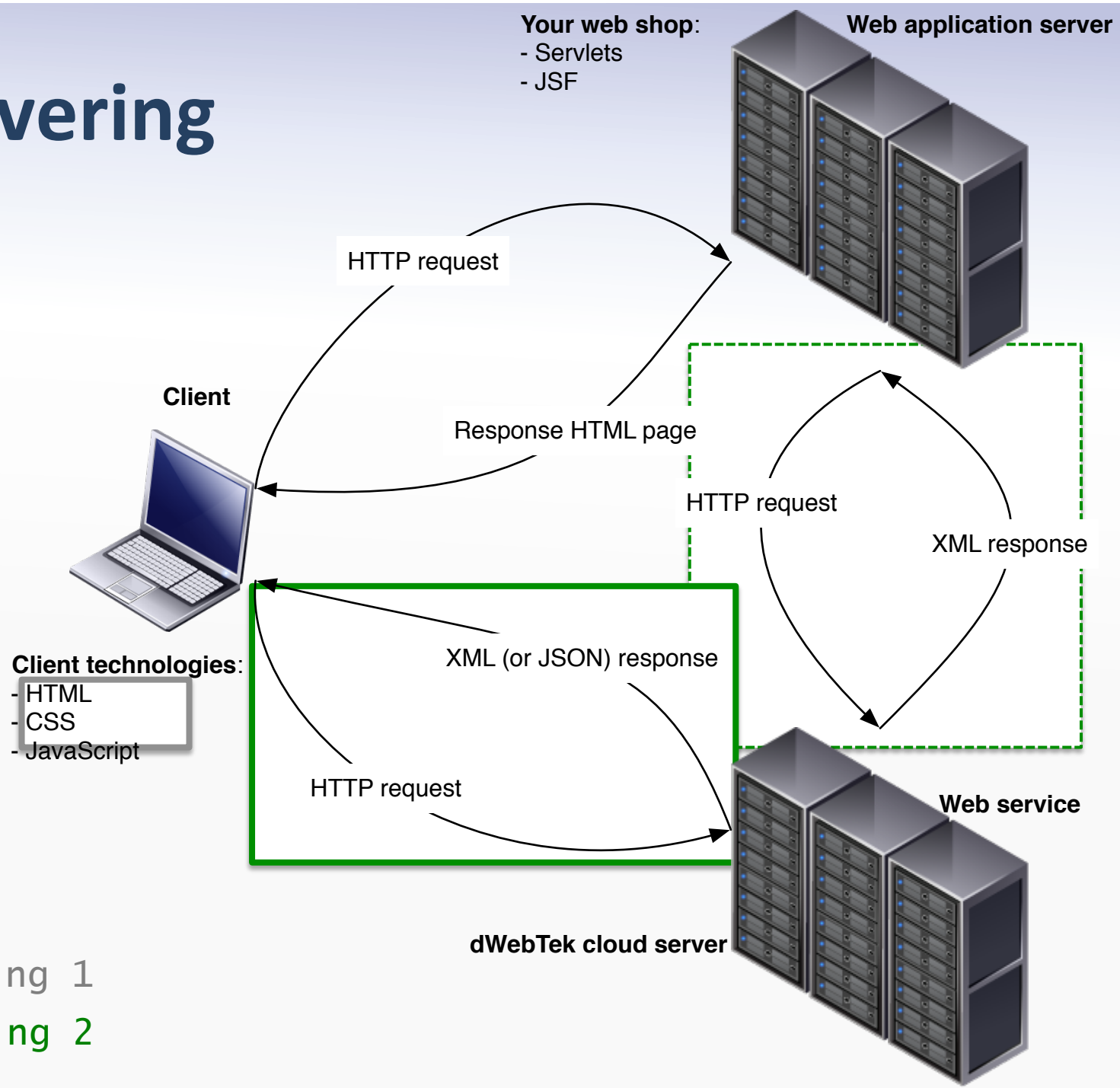
dWebTek sammenhæng



Sidste uge

Denne uge

Aflevering



Aflevering 1

Aflevering 2

Brug jeres læremestre!

dWebTek lab hver dag!