JavaScript-ohjelmointi

Yleisohjeita etäosallistumiseen

Etsi rauhallinen paikka tai mene vaikka kävelylle kuulokkeet korvilla, jos se auttaa keskittymään.

Voit tehdä omia muistiinpanoja.

Pidä mikki mykistettynä, paitsi silloin, jos haluat osallistua keskusteluun, kysyä tai kommentoida!

Voit myös kirjoittaa kysymyksiä ja kommentteja chattiin tai pyytää puheenvuoroa.

Olethan kärsivällinen, jos tulee teknisiä ongelmia puolin tai toisin.

Kurssin sisältö

1. päivä

Johdanto
- Web-standardit
- Web-sovellusten arkkitehtuurit
- Miksi JavaScript
- JavaScript versiot

Peruskäyttö
- Scriptin liittäminen sivulle
- JavaScriptin kielioppi
- Functioiden toteutus
- Muuttujien käyttö
- JavaScriptin sisäänrakennetut funktiot
- Ehtolauseet ja silmukointi

Tapahtumankäsittely
- Reagointi käyttäjän toimenpiteisiin
- Tapahtuma-olio
- Tapahtumien kupliminen
- Oletuskäsittelyn estäminen

2. päivä

Johdanto olioihin
- Mitä oliot ovat
- Oman olion toteutus
- JavaScriptin valmiit oliotyypit
- Selaimen tarjoamat oliot

Html-dokumentin muokkaus
- Elementtien etsintä
- Elementin ominaisuuksien muokkaus
- Tyylien muokkaus
- Sisällön muokkaus ja tuottaminen

Lomakkeiden käsittely
- Lomakkeen validointi
- Eri tyyppiset syöttökentät
- Syöttökenttien ominaisuuksia

Muita piirteitä
- AJAX
- Selaimen kontrollointi
- HTML5:n JavaScript APIt

Kuka

Mika Stenberg

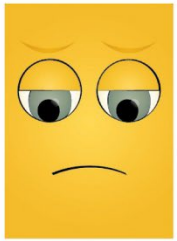
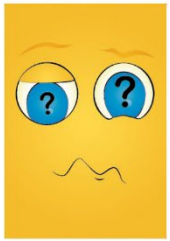
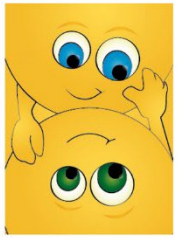
Noin. 15 v. IT-alalla ohjelmistojen ja verkkopalveluiden kehittämisen parissa. Sen jälkeen koulutushommia korkeakouluissa ja yrityksissä.

Alfame Systems, Satama Interactive, Nokia, MTV3, Almamedia Interactive, Lasipalatsin Mediakeskus, Helsingin Yliopisto, Metropolia AMK, Laurea AMK, World Aid New York

Esitietokysely

https://forms.gle/3GXsn7rc73mzb2DH8

Merkkaa kalvolle mikä fiilis päivän jälkeen?

Emoji	Happy	Sad	Confused	Inspired
				
 Bored	 Tired	 Achieved	 Angry	 Socially Involved

Kiitos osallistumisesta!

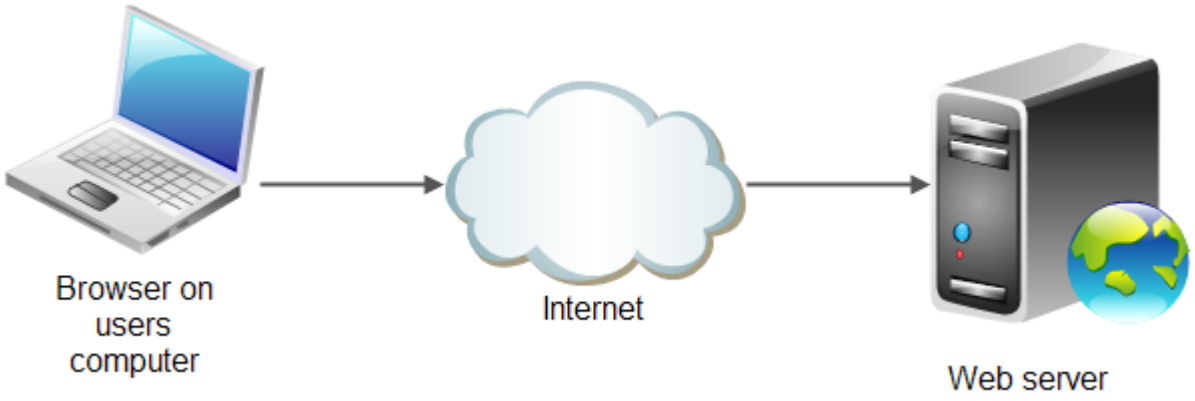
Web Development Overview

JavaScript-peruskurssi

Mika Stenberg

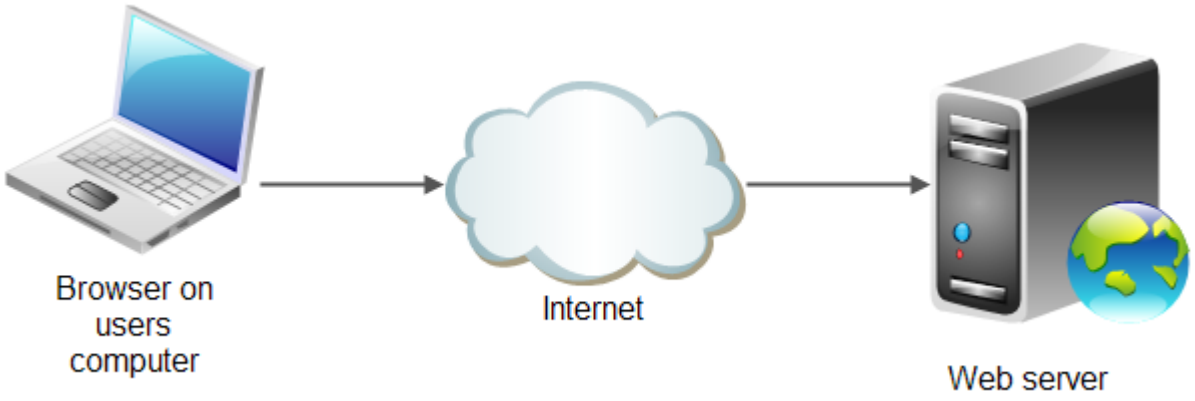
Web Development Overview

The World Wide Web



Web Development Overview

Front End Development **Back End Development**



Web Development Overview

Front End Development

PRESENTATION LAYER / FRONT-END

User interfaces

Languages: HTML, CSS, Javascript...

Frameworks: Bootstrap, Foundation 3, Grids Systems...

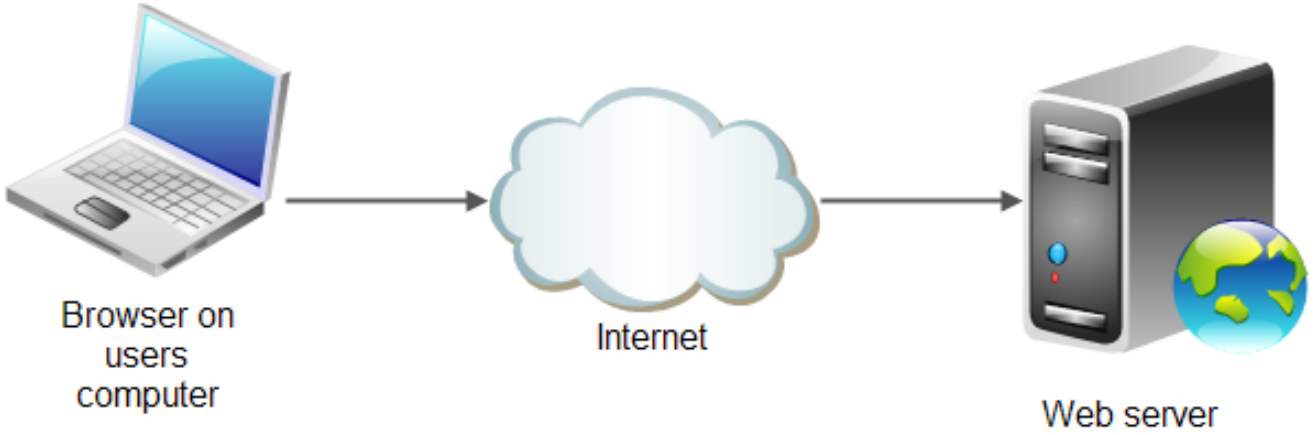
Back End Development

APPLICATION LAYER / BACK-END

Logic and operation of the website

Languages: PHP, PYTHON, RUBY, JAVA...

Frameworks: Symfony, Django, Ruby On Rails, Spring...



Web Development Overview

Front End Development Back End Development

PRESENTATION LAYER / FRONT-END

User interfaces

Languages: HTML, CSS, Javascript...
Frameworks: Bootstrap, Foundation 3, Grids Systems...

APPLICATION LAYER / BACK-END

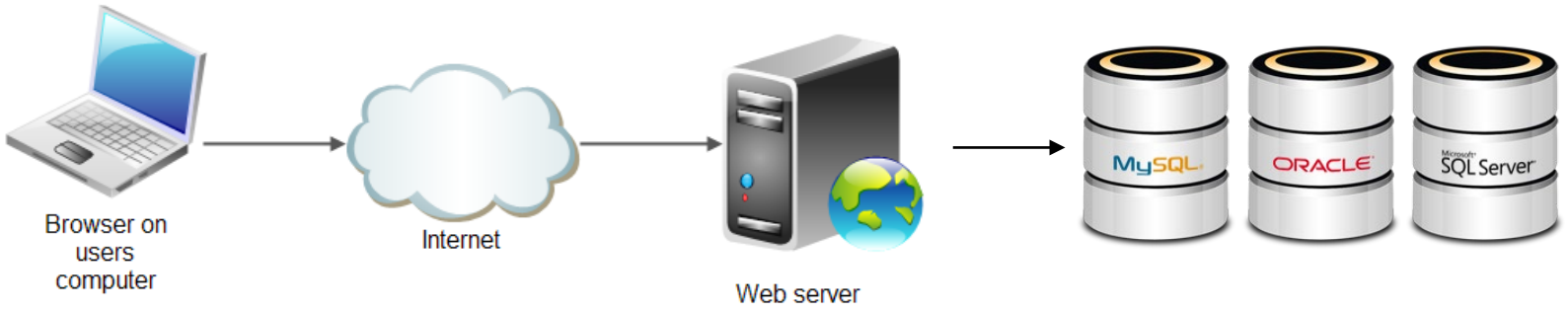
Logic and operation of the website

Languages: PHP, PYTHON, RUBY, JAVA...
Frameworks: Symfony, Django, Ruby On Rails, Spring...

DATA LAYER

Data persistence on databases

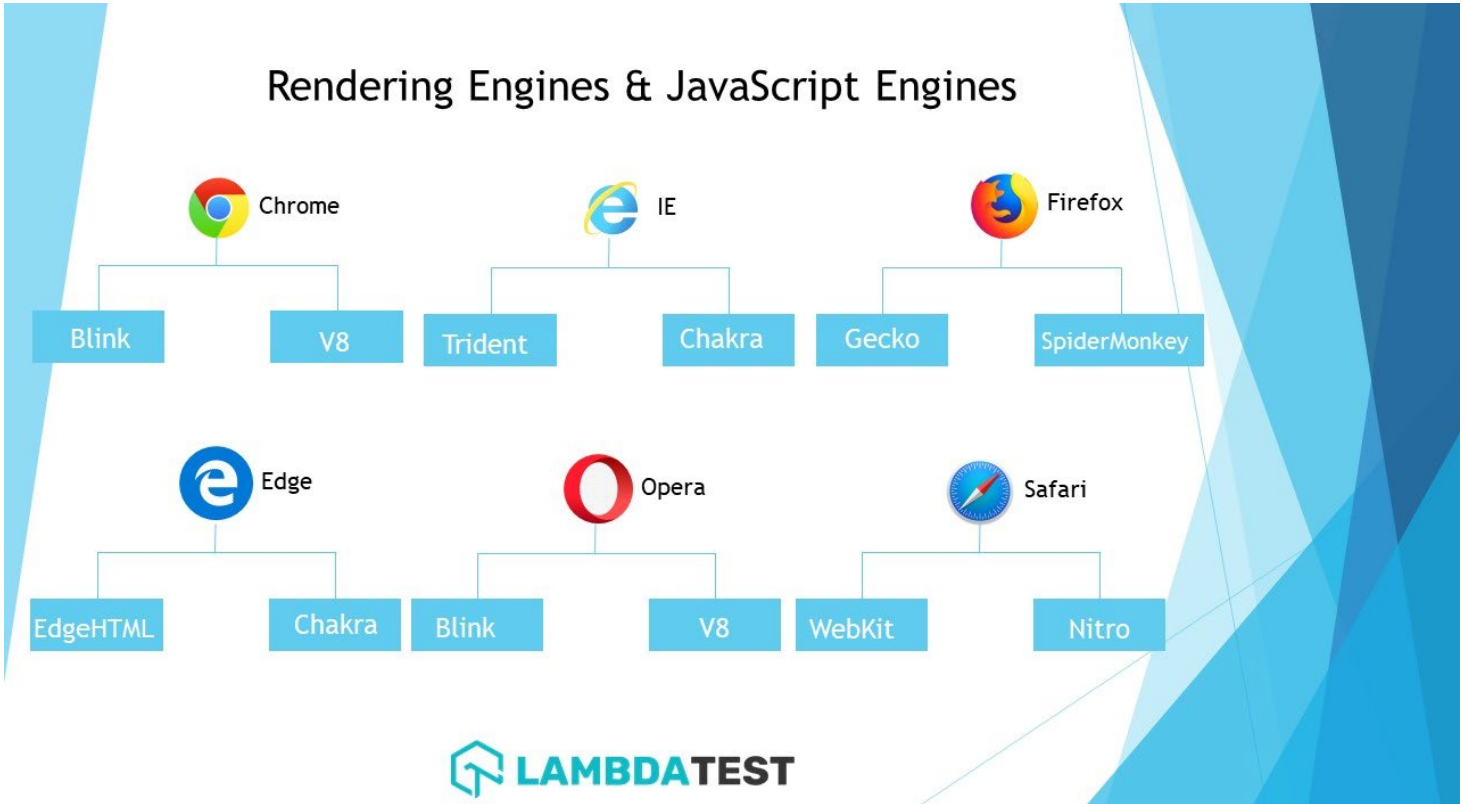
Technologies: MySQL, PostgreSQL, NoSQL...



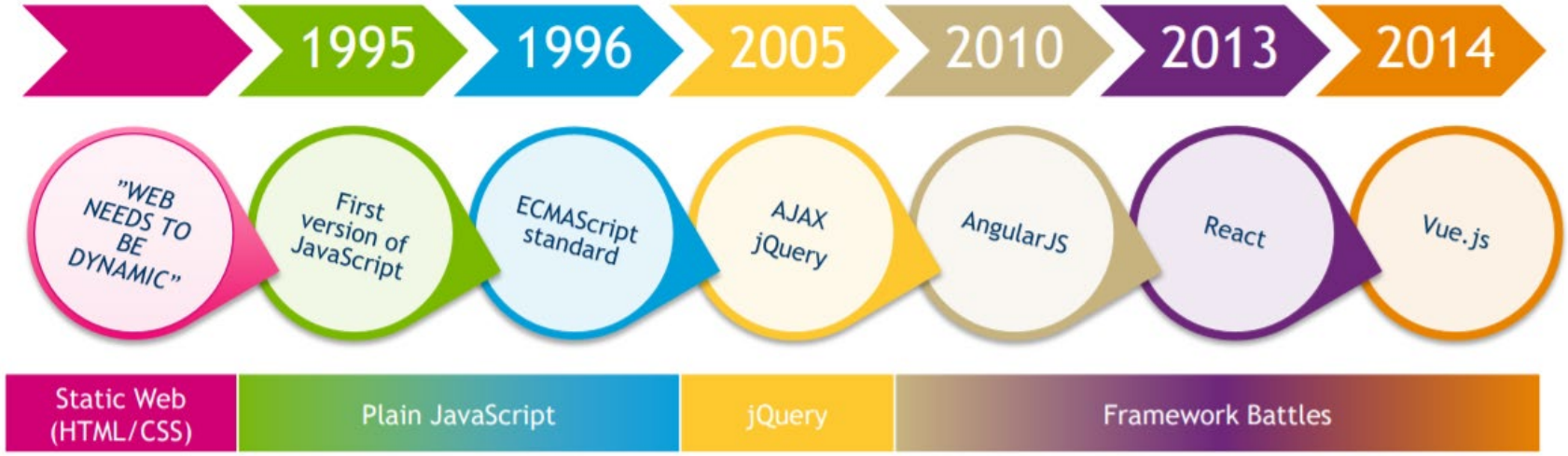
View Controller Model

Haasteet selainten kanssa

- Verkkosivut näyttävät ja/tai toimivat hieman eri tavoin eri selaimilla
- Renderöinti = Sivun esittäminen JavaScript = Koodin suorittaminen



Historia


1995
1996
2005
2010
2013
2014
"WEB NEEDS TO BE DYNAMIC"
First version of JavaScript
ECMAScript standard
AJAX jQuery
AngularJS
React
Vue.js
Static Web (HTML/CSS)
Plain JavaScript
jQuery
Framework Battles
NETSCAPE
JavaScript
ActionScript
ECMAScript
JScript
AJAX
Asynchronous Javascript And XML
jQuery

Web Development Overview

Front End Development

Back End Development

Web Development Overview

Front End Development

Bootstrap

Foundation ZURB

Pure.css

960 GRID SYSTEM

unsemantic

INK

jQuery
write less, do more.

Sass {less}

Back End Development

LAMP - stack

Linux

APACHE HTTP SERVER

MySQL

php

ME*N - stack

mongoDB

expressjs

*

NODE

Fullstack de

Front End Deve

Development

MEAN - stack

HTML CSS JS

jQu

write le

NGULA

Bootstrap

Foundation ZURB

Pure.css

960 GRID SYSTEM

unsemantic

INK

Sass {less}

APACHE HTTP SERVER

MySQL

php

mongoDB

expressjs

ANGULARJS

NODE

Kuva: Redbubbe.com

Tieturi

Fullstack JS

Front End Devel ... evelopment

PRESENTATION LAYER / FRONT-END

User interfaces

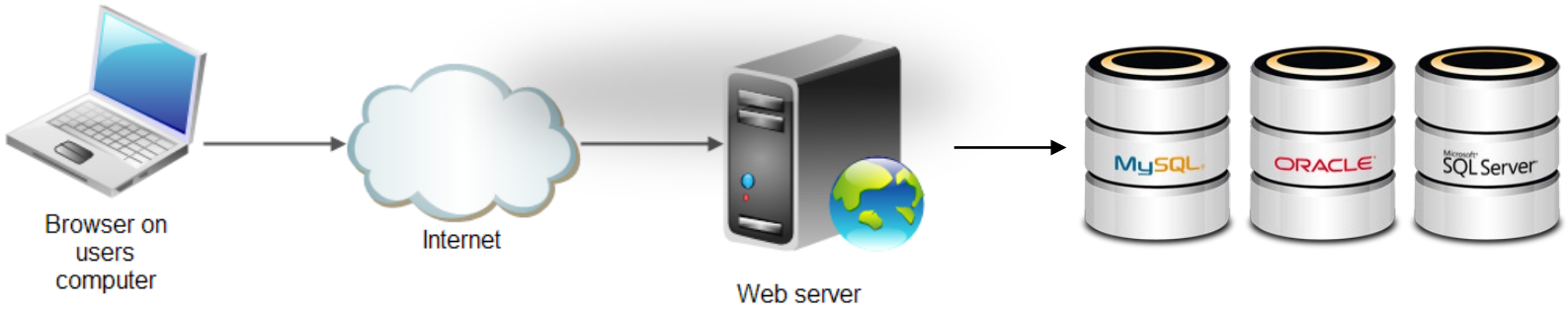
Languages: HTML, CSS, Javascript...

Frameworks: Bootstrap, Foundation 3, Grids Systems...

DATA LAYER

Data persistence on databases

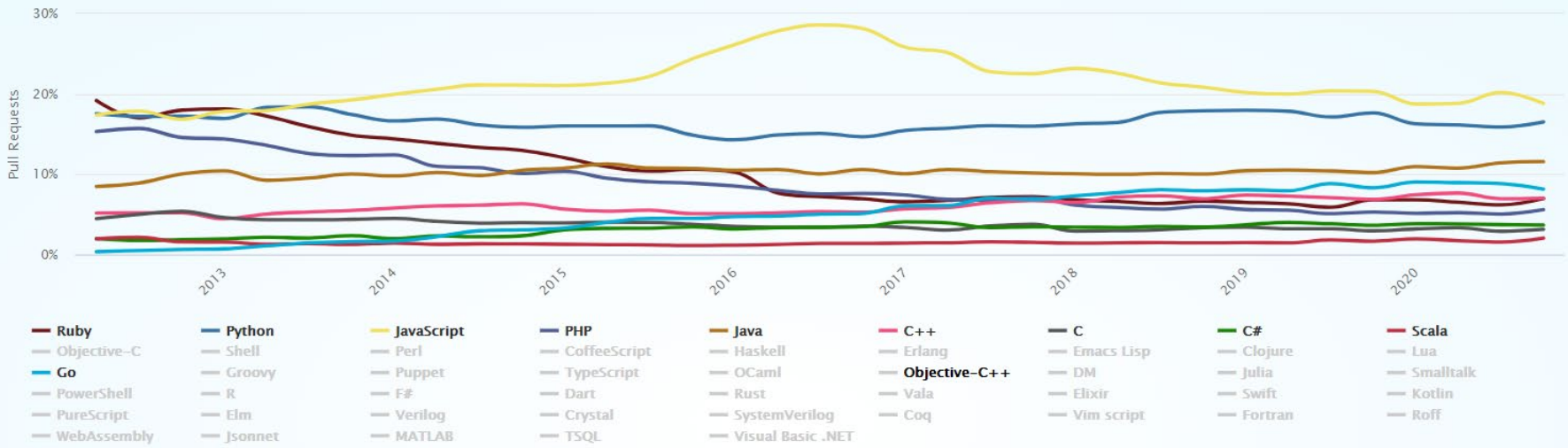
Technologies: MySQL, PostgreSQL, NoSQL...



Frontend Developer RoadMap 2020

GitHut 2.0

A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB



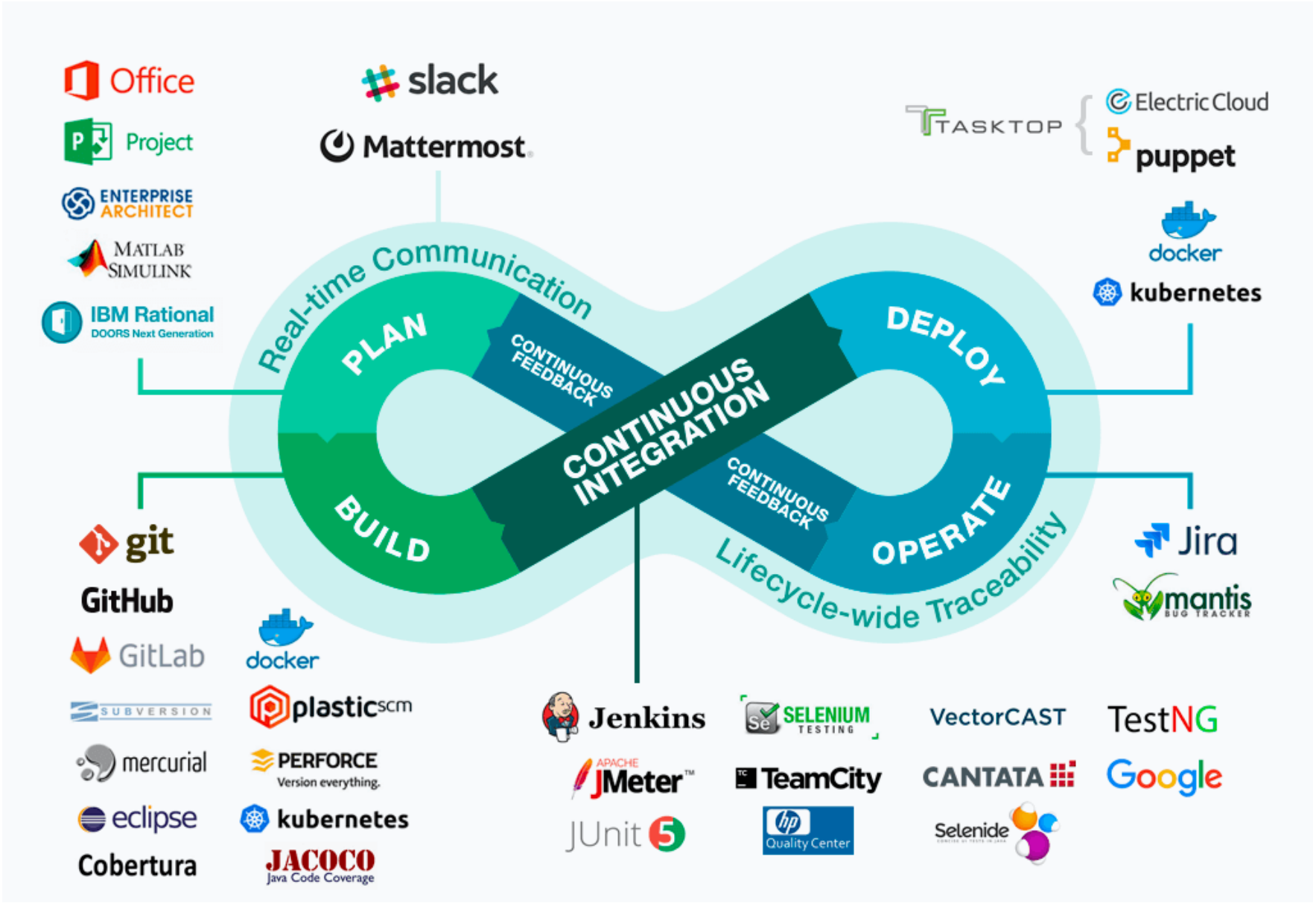
PULL REQUESTS

Year: 2020

Quarter: 4

# Ranking	Programming Language	Percentage (Change)	Trend
1	JavaScript	18.772% (-1.494%)	
2	Python	16.488% (-1.089%)	
3	Java	11.546% (+1.369%)	
4	Go	8.134% (-0.153%)	
5	C++	7.000% (+0.143%)	
6	Ruby	6.948% (+0.146%)	

https://madnight.github.io/githut/#/pull_requests/2020



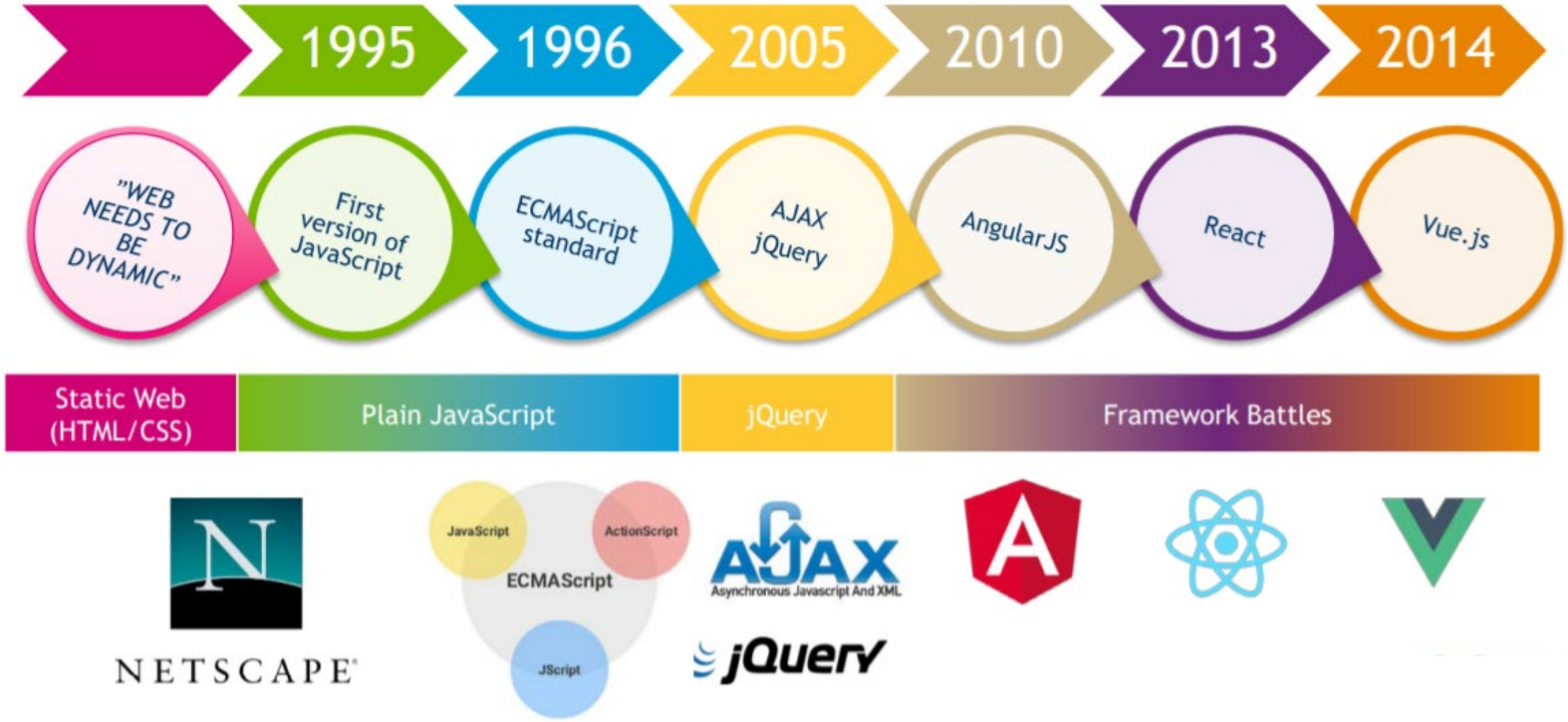
https://intland.com/codebeamer/devops-it-operations/

Introduction to JavaScript

Mika Stenberg

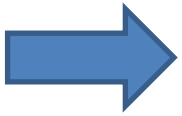
What will be discussed

1. What is JavaScript and where it came from?
2. What can be done with it?
3. How do you start coding using JS?



Where can you see it in action?

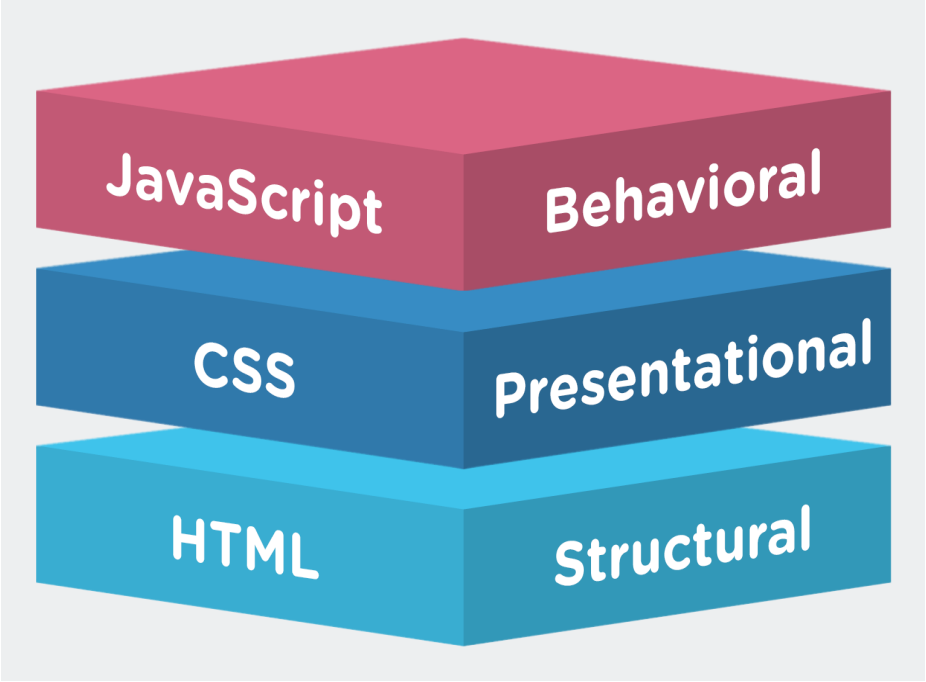
- Social media platforms: Facebook, Twitter etc.
- Google: Search Suggestions, Gmail, Maps, Analytics
- Webshops all over: shopping carts, user tracking
- Navigational menus everywhere
- Mobile apps and Desktop Apps



Wherever:

- usability & user experience counts
- content needs to be updated

Meet JavaScript!



JavaScript

```
<script type=
"text/javascript"
```

Behavior

AJAX data manipulation
Error checking
pop-up calendars
special effects

CSS

```
<style>
Body {
  Color:...
ul#mylist {
  font-family
</style>
```

Presentation

Colors
Fonts
Positioning

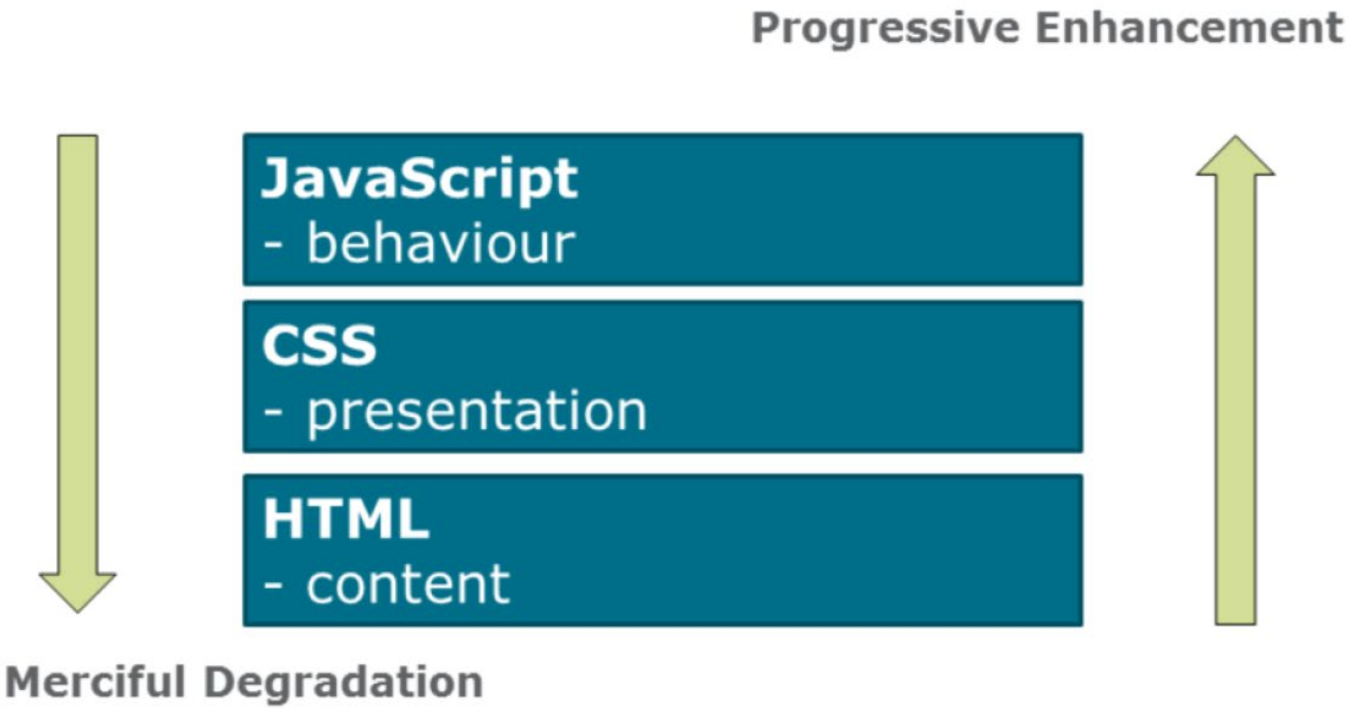
HTML

```
<html>
<head>...
<body>
<h1>
<h2>
<p>...</p>
<ol><li><li>
```

Content

Headings
Paragraphs
Lists
Images
Links

Philosophy


Progressive Enhancement
JavaScript
- behaviour
CSS
- presentation
HTML
- content
Merciful Degradation

#1. What is JavaScript

- JavaScript (JS for short) is the programming language used in web pages
- Lightweight, cross-platform, object-oriented language
- Enables page to respond to user interaction
- Is today one of the most used programming languages.
- Is used to create web applications with server side scripts (PHP, Ruby, Java) and online data sources
- **Core JavaScript** contains set of objects and fundamental language elements such as operators, control structures etc.
- **Client-Side JavaScript** extends the core providing objects to control the browser and the DOM
- **Server-Side JavaScript** provides objects relevant to running JS on a server

..and where it came from?

- Created in 1995 by Netscape as “Mocha”
- Later named as “*LiveScript*” and finally as “*JavaScript*” in 1996
- Microsoft implemented JScript in IE
- Today JS is officially specified as *ECMAScript*
  https://en.wikipedia.org/wiki/ECMAScript_version_history
- *ECMAScript versions have been abbreviated to ES1, ES2, ES3, ES5, and ES6.*
- *Since 2016, versions are named by year (ECMAScript 2016, 2017, 2018, 2019, 2020).*

Note: Browser support lags behind

- ECMAScript versions introduce new features
- Browsers implement functionalities in a delay
- See browser support for different versions:
  https://www.w3schools.com/js/js_versions.asp
  https://caniuse.com/?search=es6
- https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Browser_support_for_JavaScript_APIs
- Solution: Polyfills

A polyfill is a piece of code (usually JavaScript on the Web) used to provide modern functionality on older browsers that do not natively support it.

Babel

- Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments.

#2: What can you do with JavaScript?

- JavaScript is a ”small language” with HUGE possibilities
- Was long considered just a simple scripting utility
- Has evolved during the past 10 years to a serious challenger for traditional programming languages
- Great frameworks are available which extend the use of HTML & JavaScript drasticly

#2: What can you do with it?

1. Validate forms
2. Manipulate web pages (DOM):
   eg. add, remove, edit page contents on the fly
3. Fetch data over the web and utilize it (AJAX)
4. Store and retrieve data stored in browser (local storage)

#2: What can you do with it?

…continued

6. Develop applications, such as:

- stand-alone apps run in any browser environment
- server-side apps (Node.js)
- mobile apps (Ignite)
- Desktop apps

#3: How do you use it?

- JS code is written either in HTML-page within <SCRIPT> -tags either within <head> or <body>

```
<html>
<head>
<script>
  alert("My first App");
 </script>
</head>
```

- Or placed in a separate text-file (just like CSS-files)

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

#3: How do you use it?

- code is run when the browser encounters it on the page!
  → It makes a difference where you place your code

- <HEAD>-section is a common place to store scripts

- If code manipulates the page elements, place it before the ending </body> -tag → Why?

- We can also use onLoad –event to make sure the page has finished loading before attempting to run any scripts

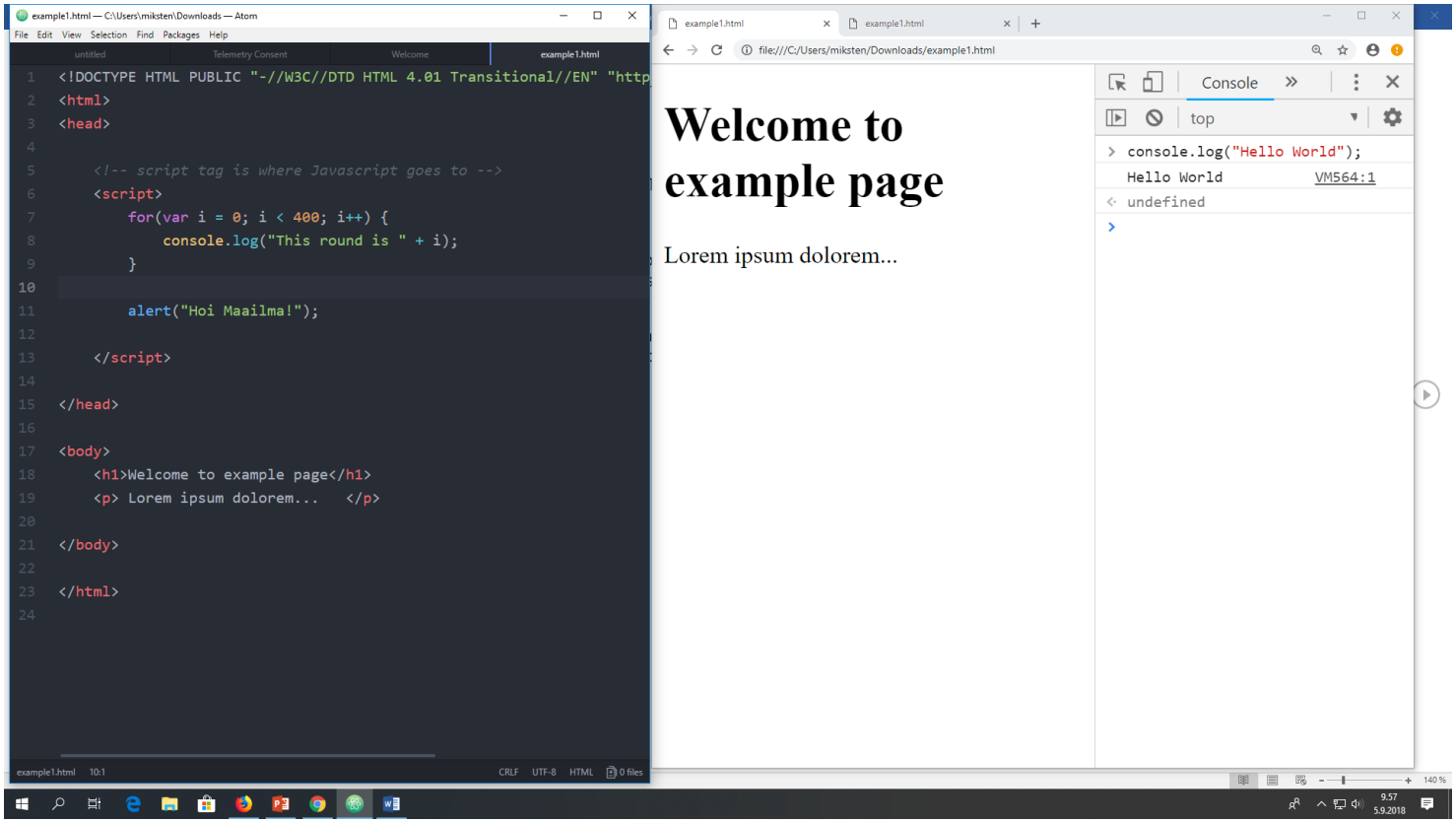
#3: Tools

- Code can be written using any text editor
- JS is run in a browser, despite the OS or device
- Code is interpreted by the browser on the fly, no compiler is needed
- Differences in browser support for JS
  Chrome and Firefox considered the fastest/stable
- Various tools are available : Chrome browser development tools and firebug add-on most used

#3: Tools used during the class

1. Visual Studio Code / Atom.io / Notepad++ (any text editor will do really!)
2. Chrome Developer Tools / Firebug add-on - offers console and debugging tools
3. Version Control (Git)
4. Cloud-based Deployment platforms, such as Netlify, Heroku etc.

Suggested screen setup



JavaScript resources

- Mozilla Developer Network
  https://developer.mozilla.org/en-US/docs/Web/JavaScript
- JavaScript in Wikipedia
  https://fi.wikipedia.org/wiki/JavaScript
- JavaScript in W3Schools
  https://www.w3schools.com/js/
- Javascript: The Definitive Guide (O'reilly)
  https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/

Questions or comments?

JavaScript BOM

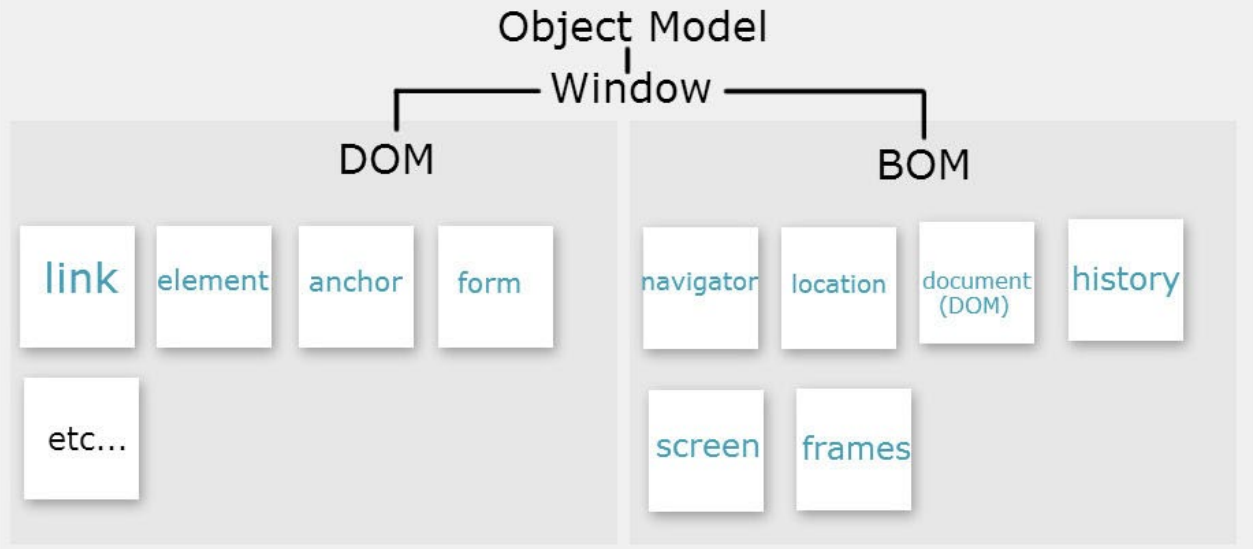
Mika Stenberg

What will be discussed

1. Talking to the browser – The BOM
2. Functions
3. JavaScript Events

1. The BOM- Browser Object Model

- Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

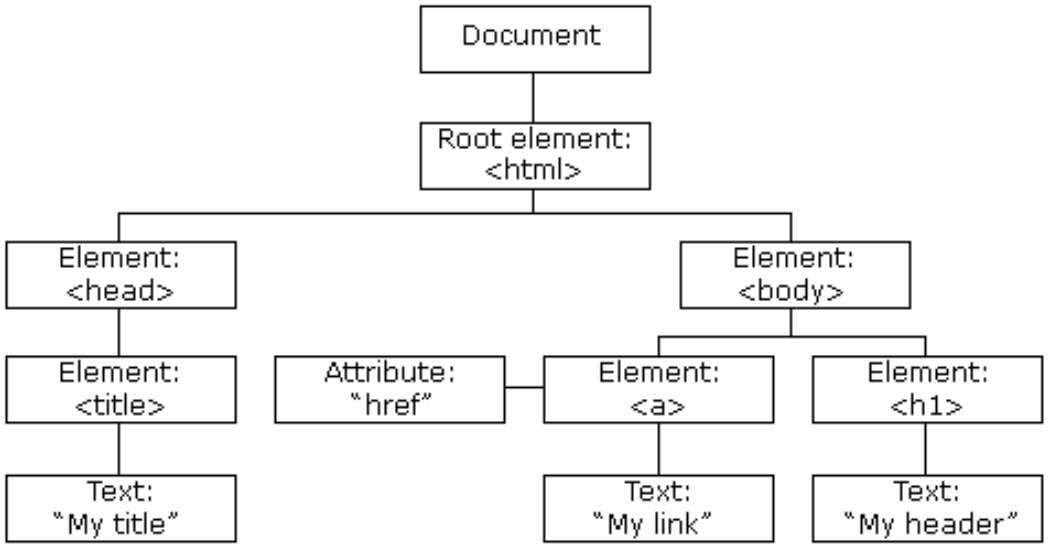


1. Using The BOM

- Allows JS to talk to the browser and get information about:
  - Browser Window contents (DOM)
  - Frames shown in page
  - Screen size, orientation, color depth
  - Navigator: browser specific information
  - History: web site history
  - Location: current web page information

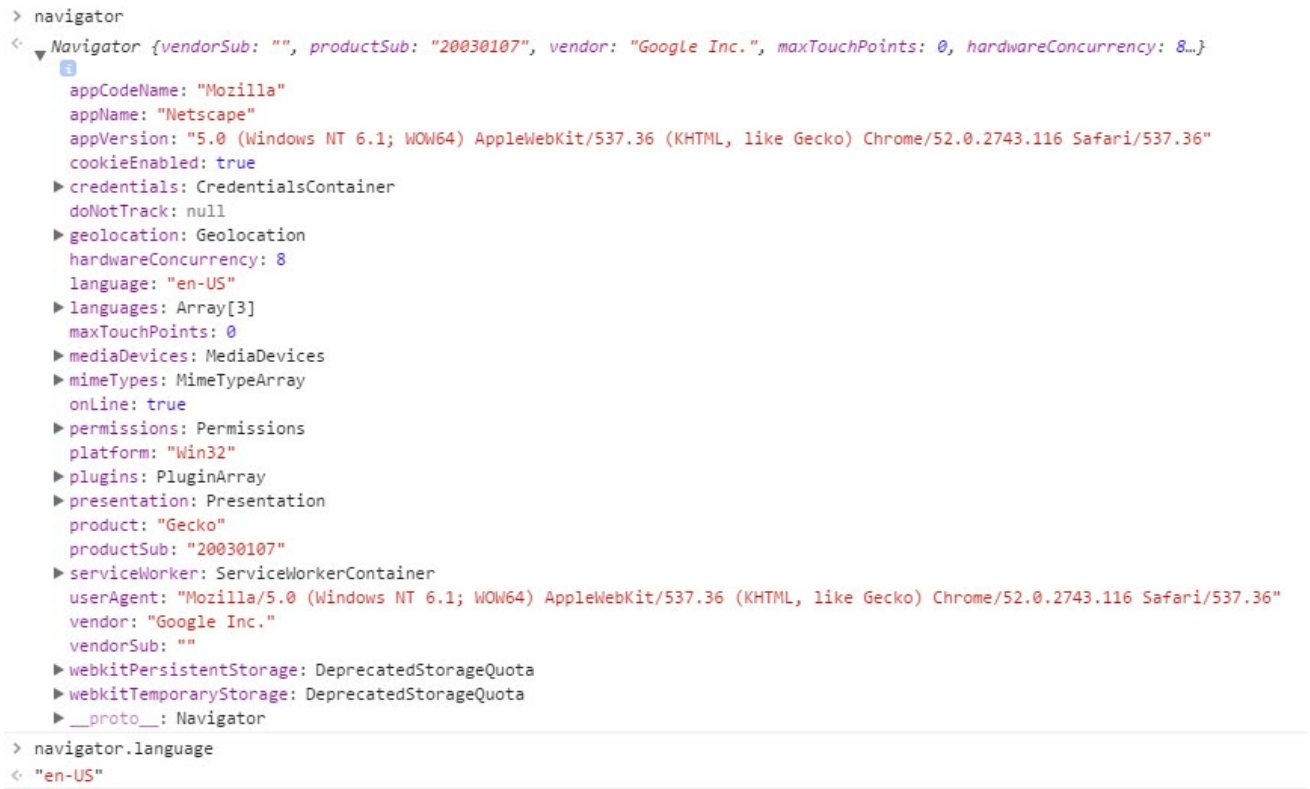
2. THE DOM – Document Object Model

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects:**



BOM: Navigator object

- For example, we can access information about the browser



BOM: Navigator object

- We can add this to the <script>-tag in the HTML page.
- Code is run when the page is loaded

```
<!DOCTYPE html>
<html>
<body>

<p>What is the name(s) of your browser?</p>

<script>
document.write("You are using: " + navigator.appName);
document.write("<br>");
document.write("Code name for the browser is " + navigator.appCodeName);
</script>

</body>
</html>

```

BOM: Window object

- Window-object lets us query the information about screen properties, such as width and height

```
> window.screen
< ▶ Screen {availWidth: 1920, availHeight: 1032, width: 1920, height: 1080, colorDepth: 24…}
> window.screen.width
< 1920
> window.screen.height
< 1080
```

BOM: History object

- History-object lets us query the information about browser history
- We can also control the browser by telling it to go back or forward in history
- NOTE: History is protected by the browser; Javascript is not allowed to read the contents of it.

```
<script>
  history.back();
</script>
```

BOM: Location object

- Location-object lets us query the information about current location
- We can also set the location which causes the browser to load it

```
> location
Location {hash: "", search: "", pathname: "/", port: "", hostname: "www.laurea.fi"…}
  ancestorOrigins: DOMStringList
  assign: function ()
  hash: ""
  host: "www.laurea.fi"
  hostname: "www.laurea.fi"
  href: "https://www.laurea.fi/"
  origin: "https://www.laurea.fi"
  pathname: "/"
  port: ""
  protocol: "https:"
  reload: function reload()
  replace: function ()
  search: ""
  toString: function toString()
  valueOf: function valueOf()
  __proto__: Location
> location.href = "http://www.iltalehti.fi";
```

Questions or comments?

JavaScript Events

Dynamic Web Apps

Introduction

- So far we've learned that the browser will run any JavaScript code whenever it encounters one while loading the page
- With an exception: code within functions will onlly be run when the function gets called
- This Chapter introduces a way to call those functions, other than from a block of code itself: events

1. HTML Events

- HTML events are "**things**" that happen to a page or its elements
- These can be something like:
  - a web page has finished loading
  - an input field was changed
  - a button was clicked
  - a form was submitted
- JavaScript can "**react**" (execute code) on these events
- Common tasks are checking or validating the input

2. Handling Events in JS

- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements

- Some examples could be:

```
<button onclick="alert('Click')">Click me</button>
<input onfocus="myFunction()"></input>
<form onsubmit="validateForm()"></form>
<button onmouseover="alert('On me!')"
onmouseout="alert('Off me')">Nada</button>
```

2. Example

- While we can write the code directly into the event, it is usually easier to call for a named function
- Then we declare the programming code later on as a function

```
<!DOCTYPE html>
<html>
<body>

<p>What is the name(s) of your browser?</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
    document.write ( "Name is " + navigator.appName);
    document.write("<br>");
    document.write ("Code name is " + navigator.appCodeName);
}
</script>

</body>
</html>
```

2. Common Events in JS

- Common events are listed below
- Full list of events can be found online

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

2. Adding listeners dynamically

- In some cases, one might want to add event listeners dynamically through JavaScript
- This can be done using **addEventListener**-method
- Removing the listener is done using **removeEventListener** –method
- Why:
  - keeps the UI and logic on a separate files and leaves HTML files clean from JavaScript
  - Separation of Concerns

2. Adding listeners dynamically

```
// Get reference to an element
var element = document.getElementsByTagName('h1')[0];
// Add
element.addEventListener("click", function(){ alert("Hello World!"); });

// Add
element.addEventListener("mouseover", function myFunction(){
        alert("Hello World!");
});


// Remove - works only on NON ANONYMOUS FUNCTIONS
element.removeEventListener("mousemove", myFunction);
```

2. Adding listeners dynamically

```
<html>

  <body>
    <h1>Eka nappi</h1>
    <p id="info">Lorem ipsum dolorem</p>
    <button id="button1">Lisää kuuntelija</button>
    <button id="button2">Poista kuuntelija</button>

      // Siirrä JS-koodi viimeiseksi ennen </body>
tägiä
   // Kaikki JS koodi, myös kuuntelijat tiedostoon
   <script src="koodit.js"></script>
  </body>
</html>
```

2. Adding listeners dynamically

```
// Etsitään viite ja lisätään kuuntelija + funktio
var x = document.getElementsByTagName('h1')[0];
x.addEventListener("click", function(){ alert("You Clicked
Me!"); });

// Etsitään viite ja lisätään kuuntelija + funktio
    var p = document.getElementById('info');
p.addEventListener("mouseover", function myFunction(){
console.log("You Clicked Me!"); });

var b1 = document.getElementById('button1');
b1.addEventListener("click", function(){
  console.log("Button clicked");
  p.addEventListener("mouseover", function myFunction(){
alert("You hovered on Me!"); });
});

// Etsitään viite ja poistetaan kuuntelija
var b2 = document.getElementById('button2');
b2.addEventListener("click", function(){
  console.log("Button clicked");
  p.removeEventListener("mouseover", myFunction, true );
});

// https://codepen.io/mjstenbe/pen/ExYJpzO
```

Finding the elements using DOM

- DOM Scripting is about finding an element and changing its attributes

Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

NOTE: new browsers also support `document.querySelector()` and `document.querySelectorAll()`

- For example:

```
var myElement  = document.getElementById('main-title');
```

Finding the elements using DOM

- Search within **the entire document** (HTML-page)

```
document.getElementById('main-navi');

// Placing the resultselt in a variable for later use
var myResult  = document.getElementById('main-navi');

// Search only within the myResult -variable
myResult.getElementsByTagName('li');
```

Questions or comments?

Smart Forms with JavaScript

Introduction

- JS was originally developed for handling forms
- With JS one can read form values, validate and check the values and react to them
- Reactions could be such as highlighting errors, providing feedback or altering the form
- As we'll come to learn, with JS one can also edit and manipulate forms (or any HTML content) freely

What will be discussed

1. The DOM - Accessing form elements & values
2. Validating form data
3. Manipulating form elements with JS
   (hightlighting, showing/hiding items)

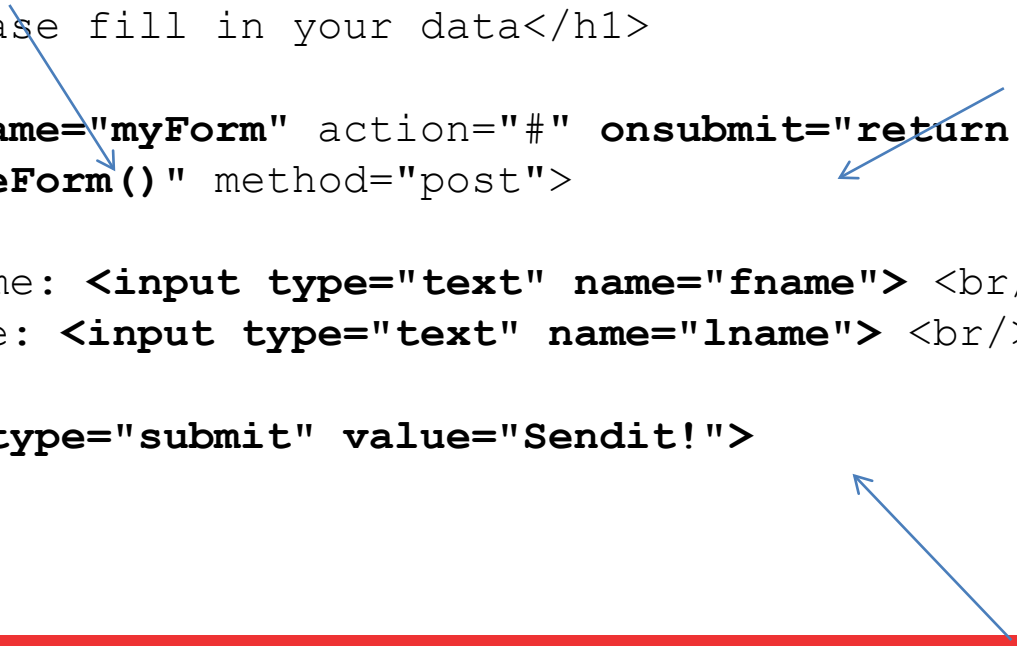
1. Accessing Form Elements

- Lets define a simple form in HTML and try it out
- It has one heading (h1), two inputs and a submit button

1. Accessing Form Elements

- Note that form elements have names and sometimes id's so we can refer to them in JavaScript
- We have also defined an event (onsubmit), which will cause the validateForm() – method to be called when the button is clicked
- NOTE: When we use **return validateForm() the form submission is cancelled if <u>false</u> is returned**

```
<h1>Please fill in your data</h1>

<form name="myForm" action="#" onsubmit="return
validateForm()" method="post">

Firstname: <input type="text" name="fname"> <br/>
Lastname: <input type="text" name="lname"> <br/>

<input type="submit" value="Sendit!">
</form>
```



1. Accessing Form Elements #1

- Accessing the elements using JS can be done in many ways, lets first look the most basic approach:

1. Since there can be multiple forms in one page, document.forms is an array, in which we have to pick the right one “myForm”’

```
var x = document.forms["myForm"]
```

2. All the form fields are stored in an array, so again we have to pick one from an array using its name “fname”

```
var x = document.forms["myForm"]["fname"]
```

1. Accessing Form Values

3. After finding the right element we can get its value using .value – field.

```
var x = document.forms["myForm"]["fname"].value
```

```
<script>
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    alert("Firstname: "+fname);
    alert("Lastname: "+lname);
}
</script>
```

1. Accessing Form Elements #2

- Another way to refer to an element is using the dot notation

```
var x = document.forms.myForm.fname.value
```

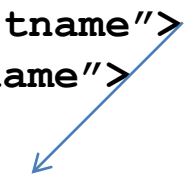
```
<script>
function validateForm() {
    var x = document.forms.myForm.fname.value;
    alert("Firstname: "+fname);
    alert("Lastname: "+lname);
}
</script>
```

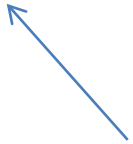
1. Accessing Form Element #3

- The easiest way to access an element, is to refer to it using its id, if such is defined in the HTML
- This is the recommended way to do things. However you may use whichever you feel more comfortable with
- We will get back to this in the coming chapters

```
var x = document.getElementById('firstname').value;
```

```
<form name="myForm" action="#" onsubmit="return
validateForm()" method="post">
Firstname: <input type="text" name="fname" id="firstname">
Lastname: <input type="text" name="lname" id="lastname">
<input type="submit" value="Sendit!">
</form>
```

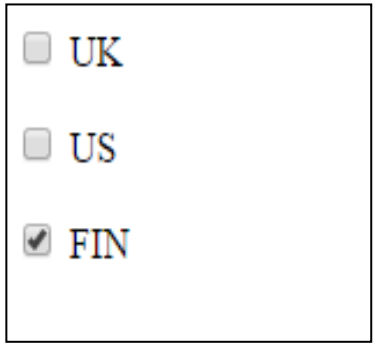




1. Accessing Checkboxes

- Different form elements behave a bit differently, when it comes to obtaining their values
- Lets see checkboxes and dropdowns for an example

```
<body>
<form id="form1" name="form1" method="post" action="">
  <p>
    <input name="MultipleOptions" type="checkbox" value="One" />
    UK </p>
  <p>
    <input name="MultipleOptions2" type="checkbox" value="Two" />
    US </p>
  <p>
    <input name="MultipleOptions3" type="checkbox" value="Three" checked="checked" />
    FIN </p>
</form>
</body>
```

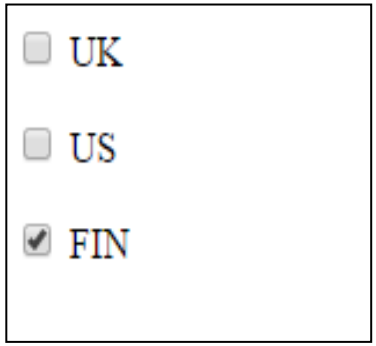


1. Accessing Checkboxes

- Getting the data for these would go like:

```
document.forms.form1.MultipleOptions3.value;
>> "Three"
document.forms.form1.MultipleOptions3.checked;
>> "True"
```

```
<body>
<form id="form1" name="form1" method="post" action="">
  <p>
    <input name="MultipleOptions" type="checkbox" value="One" />
    UK </p>
  <p>
    <input name="MultipleOptions2" type="checkbox" value="Two" />
    US </p>
  <p>
    <input name="MultipleOptions3" type="checkbox" value="Three" checked="checked" />
    FIN </p>
</form>
</body>
```



1. Accessing Dropdowns

- `document.forms.form1.MyMenu.options.length;`
  `>> 3`
- `document.forms.form1.MyMenu.options;`
  `>>[<option>FIN</option>, <option>UK</option>, <option>US</option>]`
- `document.forms.form1.MyMenu.options[1].value;`
  `>> "UK"`

```
<form id="form1" name="form1" method="post" action="">
  <select name="MyMenu" size="1">
    <option>FIN</option>
    <option>UK</option>
    <option>US</option>
  </select>
```

2. Validating the data

- After getting the data, it's easy to validate it

```
<script>
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == null || x == "") || x.length < 3) {
        alert("Name must be filled out");
        return false;
    }
}
</script>
```

2. Validating the data

- Simple validation can actually be done iusing HTML 5 attributes if the client is using a modern browser
- See more online at: http://www.the-art-of-web.com/html/html5-form-validation/

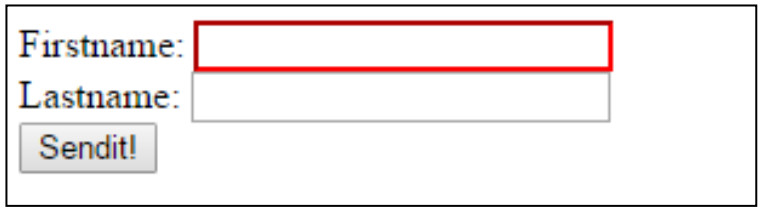
```
<h1>Please fill in your data</h1>

<form name="myForm" action="#" onsubmit="return
validateForm()" method="post">

Firstname: <input type="text" name="fname" required> <br/>
Lastname: <input type="text" name="lname" min=5> <br/>

<input type="submit" value="Sendit!">
</form>
```

3. Manipulating the form

- It is common to have the form react to user actions in one way or another
- Commos ways are:
  - highlight errors
  - display error messages
  - focus on the field
  - or toggle visibility of fields based on earlier input
- Lets research into these for a while

3. Manipulating the form #1

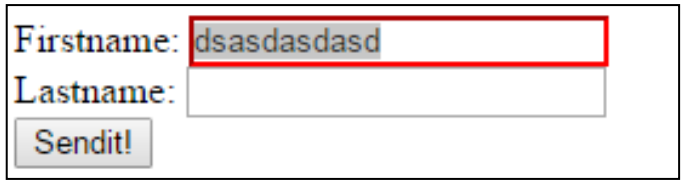
- Highlighting an element is one common approach for errors
- This can be done by referring to an element in a form and altering its style using JavaScript




```
// highlight the field
document.forms.myForm.fname.style.borderColor = "red";

// or by using a variable to hold the reference and an ID:

var x = document.forms.myForm.fname;
x. style.borderColor = "red";
```

3. Manipulating the form #2

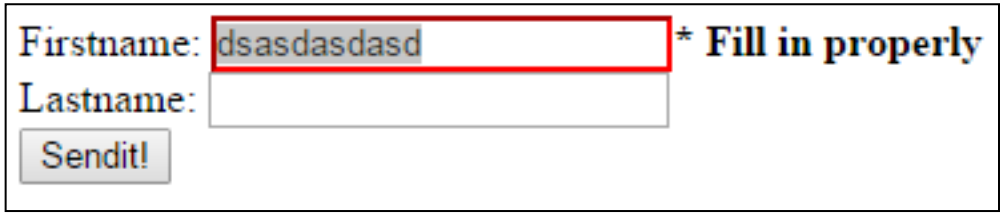
- Furthermore we could select the erroneous input
- And focus on the field




```
// highlight the field
var x = document.forms.myForm.fname;
x. style.borderColor = "red";

// select all text and set mouse focus on the field
x.select();
x.focus();
```

3. Manipulating the form #3

- And finally we could add a notification

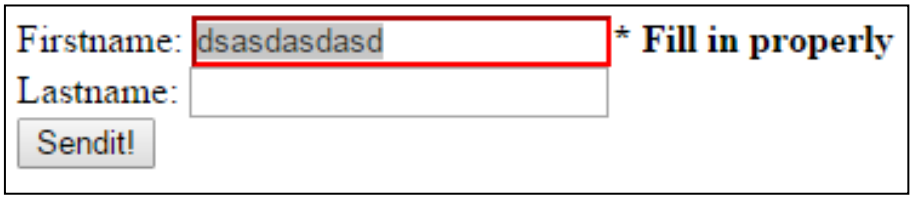


- We need to add span-element next to input, in which we can then add our text

```
<input type="text" name="fname" id="fname">
<span id="feedback"></span> <br/>
```

3. Manipulating the form #3

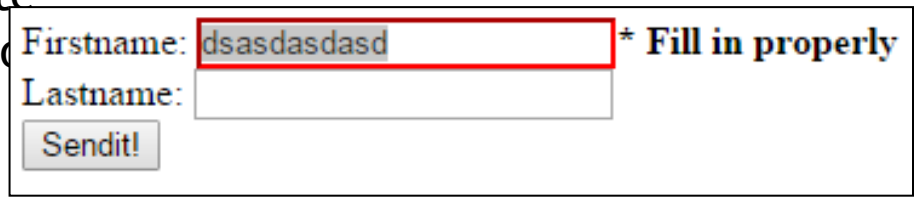
- Adding HTML and text into an element can be done by editing innerHTML – field of any element.




```
// give feedback (quick & dirty)
document.getElementById('feedback').innerHTML=”<b>*Fill in properly</b>";
```

3. Manipulating the form #3

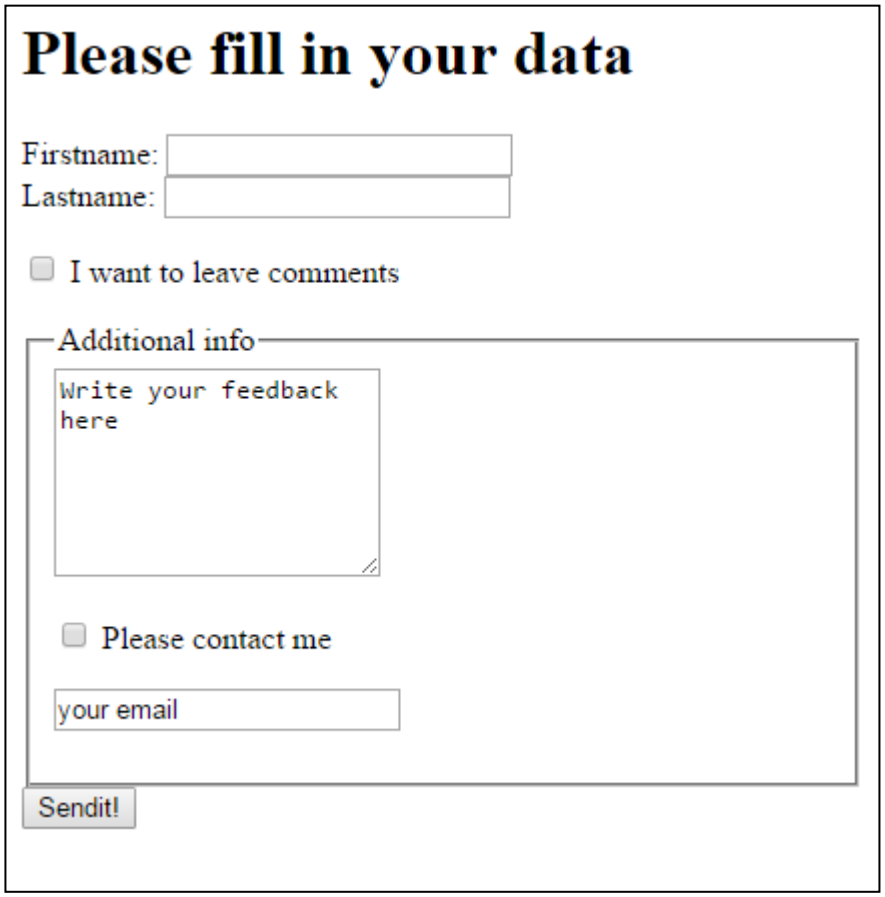
- We can also dynamically create the elements and add them to the page
- This results the same HTML




```
 // Create a <b> element (bold text)
var h = document.createElement("b") ;

 // Create a text node
var t = document.createTextNode(”* Please fill in properly");

// Attach one to another
h.appendChild(t);

// Finally add the created nodes to feedback element
 document.getElementById('feedback').appendChild(h);
```

3. Manipulating the form #4

- Last case is toggling the visibility of items based on user input
- This is done by setting elements visibility attribute on/off
- Lets start out with a form
- Additional info will be hidden, and shown only if the user ticks on the first checkbox



3. Manipulating the form #2

```
<form name="myForm" action="#" onsubmit="#" method="post">
  <p> Firstname:
    <input type="text" name="fname" id="fname">
    <span id="feedback"></span> <br/>
    Lastname:
    <input type="text" name="lname" id="lname">
  </p>
  <p>
    <input type="checkbox" name="comments" id="comments">
    I want to leave comments</p>
  <p>
  <fieldset>
    <legend>Additional info</legend>
    <textarea name="message" id="message">Write your feedback here
      </textarea>
    </p>
    <p>
      <input type="checkbox" name="contactme" id="contactme">
      Please contact me</p>
    <p>
      <label for="email"></label>
      <input name="email" type="text" id="email" value="your email">
      <br/>
  </fieldset>
  <input type="submit" value="Sendit!">
  </p>
</form>
```

3. Manipulating the form #2

1. Lets first hide the additional fieldset
   (NOTE: if JS is disabled, then extra fields will still show up).

```
var x = document.getElementById('extraFields');
x.style.display = "none";
```

2. The we add an onClick -event to the checkboxs HTML , which will call a funtion and change the visibility either on or off

```
<input type="checkbox" name="comments" id="comments" onClick="showExtraFields()">
I want to leave comments</p>
```

3. Manipulating the form #2

```
function showExtraFields(){

// get references to elements
var extraFieldset = document.getElementById('extraFields');
var checkBox = document.forms.myForm.comments.checked;

// if the box is not checked when user clicks it, hide fields
if (!checkBox){
  extraFieldset.style.display = "none";
} // otherwise, show them
else {
 extraFieldset.style.display = "block";
}

}
```

Questions or comments?

AJAX

Mika Stenberg

JavaScript

Nieturi

What will be discussed

1. What is AJAX
2. How it works?
3. Creating a request
4. Handling the response
5. Browser Considerations
6. Callback functions
7. Use Case Scenarios

1. What is AJAX

- AJAX = Asynchronous JavaScript and XML.
- AJAX **is not a new programming language**, but a new way to use existing standards.
- AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.
- Made famous by Google Search Suggestions
- Imagine: an HTML table; updating the data without reloading the page

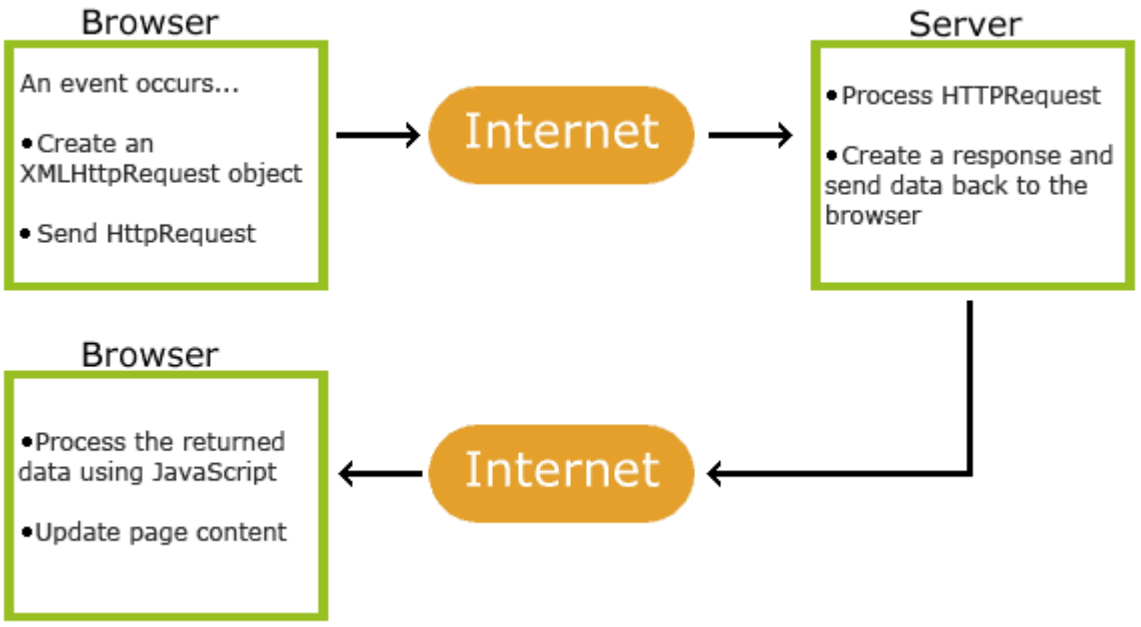
1. What is AJAX

- AJAX is about updating parts of a web page, without reloading the whole page
- web pages are updated asynchronously by exchanging small amounts of data with the server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.
- Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

1. Use Case Scenarios

- Add / delete a row of data in a table, update the database on the background without reload
- Fetch data from News Service, display to the user
- Fetch data from weather service as xml, parse and display
- Google: search database and suggest
- Facebook, Gmail, Google Maps:
  constant update of site data without reload
- Try out some:
  http://www.w3schools.com/ajax/ajax_examples.asp

2. How it works



W3School: Ajax

2. How it works

- AJAX is based on internet standards, and uses a combination of:
  - JavaScript/DOM (to display/update the page)
  - CSS (to style the data)
  - XML or JSON (format for transferring data)
  - XMLHttpRequest object (to exchange data asynchronously with a server)

2. How it works

- Sounds more complicated than it is!
- Basically there are three steps in every AJAX program

1. Creating and sending the AJAX request
2. Receiving the response
3. Parsing and displaying the response

2. How it works

- Sounds more complicated than it is!
- Basically there are three steps in every AJAX program

1. Creating and sending the AJAX request
   - XMLHttpRequest object (to exchange data)

2. Receiving the response

3. Parsing and displaying the response
   - XML or JSON (format for transferring data)
   - JavaScript/DOM (to display/update the page)
   - CSS (to style the data)

3. Creating a request

- We’ve already covered most of the techniques, there’s only XMLHttpRequest to cover
- Creating an XMLHttpRequest object is easy:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET",http://www.mysite.org/demo_get.php",true);
xmlhttp.send();
```

3. Creating a request

- Sounds more complicated than it is!
- We've already covered most of the techniques, there's only XMLHttpRequest to cover
- Creating an XMLHttpRequest object is easy:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET",somefile.txt",true);
xmlhttp.send();
```

Can be either a call to a script, which will return some data

OR

Static file which's contents will be returned

4. Handling the response

- The code we wrote send the request
- We still have to specify what to do with the response

```
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4 &&
xmlhttp.status==200){
   document.getElementById("myDiv").innerHTML=
    xmlhttp.responseText;
    }
}
```

4. Handling the response

- The code we wrote send the request
- We still have to specify what to do with the response

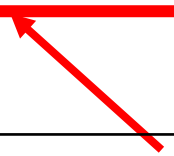
```
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4 &&
xmlhttp.status==200){
   document.getElementById("myDiv").innerHTML=
    xmlhttp.responseText;
    }
}
```

Add an ”onreadystatechange” event listener to xmlhttp. When we receive the response, browser will run the function

4. Handling the response

- The code we wrote send the request
- We still have to specify what to do with the response

```
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4 &&
xmlhttp.status==200){
   document.getElementById("myDiv").innerHTML=
    xmlhttp.responseText;
    }
}
```



Set the contents of ”myDiv” element to the received response

4. Handling the response

- OnReadyStateChange –object has a set of values we can respond to

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

4. AJAX the new way - Fetch API

```
fetch(url)
  .then((response) => {
    return response.text();
  })
  .then((data) => {
    // do something with 'data'
  });
```

See: https://www.codeinwp.com/blog/fetch-api-tutorial-for-beginners/

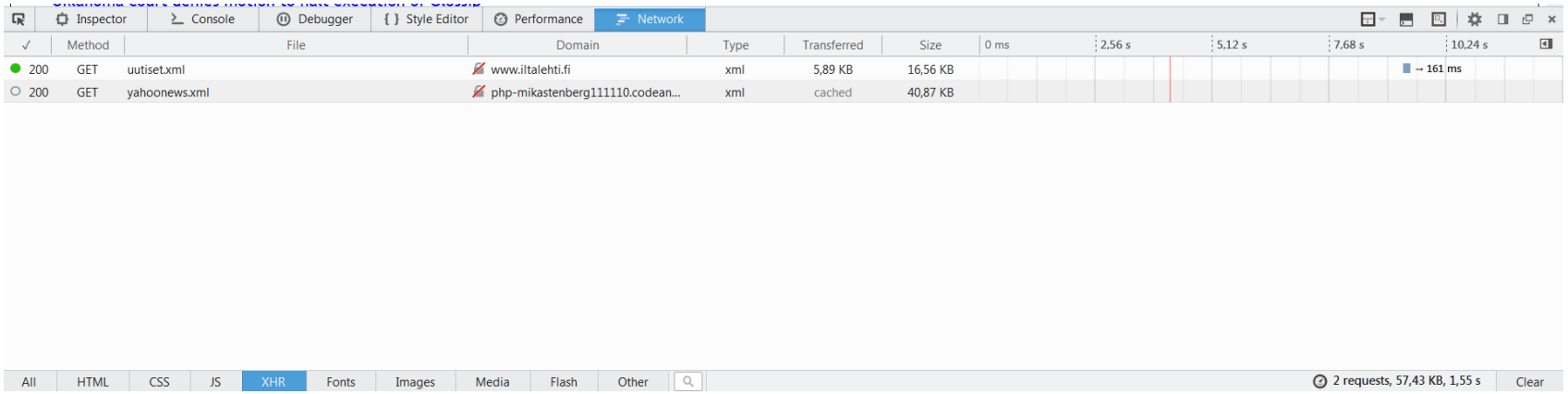
4. Contents of the response

- The respond is stored in **xmlhttp.responseText** - variable
- The respond we receive can contain any data
- Usually it is one of the following:

1. Plain text or HTML
2. XML formatted data
3. JSON formatted data

Developer Tools and XHR

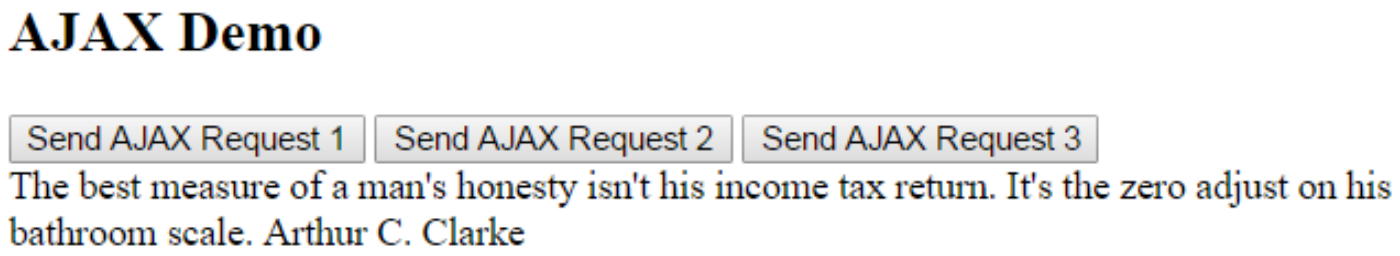
- Developer Tools will help you track and debug Ajax (XHR) calls
- One can see the status, size, response time and raw content of the queries



4. Handling the contents: Plain text / HTML

1. Plain text or HTML
   - Usually just added to the page as it is
   - Can be examined and manipulated if needed
   - For example:

```
document.getElementById("myDiv").innerHTML= xmlhttp.reponseText;
```



4. Handling the contents: XML data

- RAW data which needs to be processed
- We can add styles & structure
- Pick the data fields using getElements ByTagName() – function or
- using for -loops

```
<contents>
    <quotes>
        <quote>I'm not concerned about all hell breaking
t.</quote>
        <length>124</length>
        <author>George Carlin</author>
        <tags>funny</tags>
        <tags>hell</tags>
        <tags>humor</tags>
        <tags>insightful</tags>
        <category>funny</category>
        <id>MTIsw02XdSMG267H1qNiFAeF</id>
    </quotes>
```

I'm not concerned about all hell breaking loose, but that a PART of hell will break loose... it'll be much harder to detect.	George Carlin
The biggest problem with every art is by the use of appearance to create a loftier reality.	Johann Wolfgang von Goethe

4. Handling the contents: XML data

- RAW XML data needs to be processed, makes no sense displaying it directly
- We can select the data fields using getElements ByTagName() -function
- When we output the values, we can add structure (using HTML) and style (using CSS) like below

```
<contents>
    <quotes>
        <quote>I'm not concerned about all hell breaking
t.</quote>
        <length>124</length>
        <author>George Carlin</author>
        <tags>hell</tags>
        <tags>humor</tags>
        <tags>insightful</tags>
        <category>funny</category>
        <id>MTIsw02XdSMG267H1qNiFAeF</id>
    </quotes>
```

I'm not concerned about all hell breaking loose, but that a PART of hell will break loose... it'll be much harder to detect.	George Carlin
The biggest problem with every art is by the use of appearance to create a loftier reality.	Johann Wolfgang von Goethe

4. Parsing XML data for single elements

```
xmlhttp.onreadystatechange=function() {
 if (xmlhttp.readyState==4 && xmlhttp.status==200){
// Save the response data in a variable for easy processing
   var xmlDoc = xmlhttp.responseXML;

// Use getElementsByTagName to dig out quote-tags (note that it is an array!)
var quotes =  xmlDoc.getElementsByTagName("quote");
var firstQuote = quotes[0];
Var quoteText = firstQuote.innerHTML;

// Finally we will place the information on screen
   document.getElementById("myDiv").innerHTML= quote;
    }
}
```

- See more on this at: http://www.w3schools.com/xml/dom_intro.asp

4. Parsing XML data for multiple elements

- Sometimes XML data contains multiple items, such as ”quotes” in the image
- Searching the element tags returns an array of item
- We use for-loop to iterate through all the items in the array
- For each item, we do the same thing that we did in the previous slide; select the spesific tags we want to extract

4. Parsing XML data for multiple elements

- Sometimes XML data contains multiple items, such as ”quotes” in the image
- **Searching the XML tags such as ”quotes” returns an array of items**
- We use for-loop to iterate through all the items in the array
- For each item, we do the same thing that we did in the previous slide; select the spesific tags we want to extract

4. Parsing XML data for multiple elements

- Sometimes XML data contains multiple items, such as ”quotes” in the image
- Searching the XML tags such as ”quotes” returns an array of items
- We use for-loop to iterate through all the items in the array
- **For each item, we do the same thing that we did in the previous slide; select the spesific tags we want to extract**

4. Parsing XML data for multiple elements

- Sometimes XML data contains multiple items, such as ”quotes” in the image
- Searching the XML tags such as ”quotes” returns an array of items
- We use for-loop to iterate through all the items in the array
- **For each item, we do the same thing that we did in the previous slide; select the spesific tags we want to extract**

4. Parsing XML data for multiple elements

```
xmlhttp.onreadystatechange=function() {
 if (xmlhttp.readyState==4 && xmlhttp.status==200){
// Save the response data in a variable for easy processing
   var xmlDoc = xmlhttp.responseXML;

// Use getElementsByTagName to dig out quote-tags (note that it is an array!)
var allquotes =  xmlDoc.getElementsByTagName("quote");

// Use for-loops to iterate the array of quotes
for (i = 0; i< allquotes.length; i++) {
    quote = allquotes[i].childNodes[0].nodeValue + "<br>";
    document.getElementById("myDiv").innerHTML += quote;
}
```

4. XML attributes

- Sometimes XML data elements can contain attributes

```
▼<wind>
   <speed value="5.26" name="Gentle Breeze"/>
   <gusts/>
   <direction value="201.003" code="SSW" name="South-southwest"/>
 </wind>
```

- Value of the attributes can be retrieved using getAttribute() -function

```
var element = xmldoc.getElementsByTagName('speed')
var speed = element[0].getAttribute('value');
```

4. Parsing XML data for attributes

```
xmlhttp.onreadystatechange=function() {
 if (xmlhttp.readyState==4 && xmlhttp.status==200){
// Save the response data in a variable for easy processing
   var xmlDoc = xmlhttp.responseXML;

// Use getElementsByTagName to dig out quote-tags (note that it is an array!)
var allquotes =  xmlDoc.getElementsByTagName("quote");

// Use for-loops to iterate the array of quotes
for (i = 0; i< allquotes.length; i++) {
    quote = allquotes[i].innerHTML + "<br>";
    document.getElementById("myDiv").innerHTML += quote;
}
```

4. Handling the contents: JSON

- =JavaScript Object Notation
- Up to 10x faster to parse than XML
- Much smaller in size
- Might be easier to parse

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

4. Parsing JSON data

- Raw Data:

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

- Variable is used in the code as an array or an object, depending on the structure

obj.employees[1].firstName
obj.employees[1].lastName;

- More on this at: http://www.w3schools.com/json/

5. Browser Considerations

- Older browsers (IE) have a different way of creating the XMLHttpRequest –object

```
// code for IE7+, Firefox, Chrome, Opera,
Safari
if (window.XMLHttpRequest){
    xmlhttp = new XMLHttpRequest();
}
// code for IE6, IE5
else {
    xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");
}
```

Questions or comments?

Storing data, working offline

Mika Stenberg

JavaScript

Tieturi

What will be discussed

1. Saving data locally
   - Cookies
   - Localstorage and SessionStorage
   - WebSQL
   - IndexedDB
2. Working offline
   - Utilizing AppCache

Why Save Data?

- When the user loads another page, the browser doesn’t know what he/she has done
- Saving data makes it possible to store user sessions!
- For example: Shopping carts, log-in data, user history

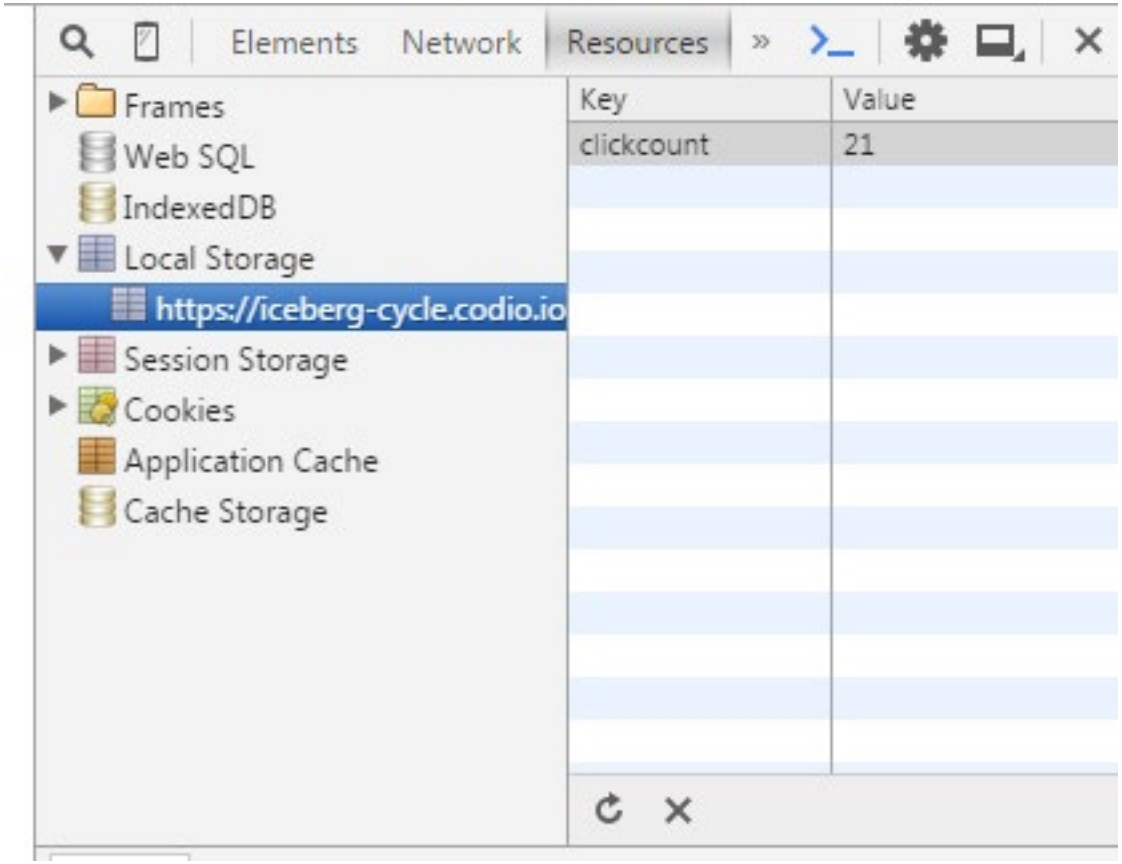
2.Saving Data Locally

- We have several ways of saving data in browser

1. Cookies
2. LocalStorage and SessionStorage (HTML5)
3. IndexedDB (HTML5)
4. WebSQL (Deprecated) (HTML5)

2. Saving data

- You can use browser Developer Tools to explore the saved data



2. Cookies

- Cookies are the oldest way of saving data in browser
- As a default cookies are removed when the browser closes, unless we set an expiry date
- Setting a cookie with JS is simple:

```
document.cookie="username=John Doe";
```

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

2. Reading cookies

- Reading a cookies

```
var x = document.cookie;
```

- To change a cookie, we need to re-set it

- Cookie will be removed when it reaches the expiry date. We can also remove it by setting the date to past

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC";
```

2. Cookie disadvantages

- Insecure – everythin is stored in plaintext.
- Can store only 4 kb's of data
- Easy to view/hack just browser tools
- Cookies are sent unencrypted to the server as well

2. Localstorage

- Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Local storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

2. Localstorage

- Part of the HTML5 specification
- Not supported by all browsers
-

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

```
if(typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

2. Localstorage

- Using localstorage

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

- Removing an item

```
localStorage.removeItem("lastname");
```

2. Localstorage with Complex Data

- Localstorage can only save String-values!

```
var taulu = [1,2,3,4,5,6,7];
localStorage.setItem(taulukko, taulu); // String
localStorage.getItem("taulukko");
>"1,2,3,4,5,6,7”// returned as String, not array
```

- Workaround! Save data as JSON string

```
var taulu = [1,2,3,4,5,6,7];
var merkkijonotaulu = JSON.stringify(taulu); // to string
JSON.parse(merkkijonotaulu,"", " "); // to array
(7) [1, 2, 3, 4, 5, 6, 7] // string is converted to array

var auto = {"vuosi" : 2000, "Merkki": "Pösö"};
var autoMerkkijono = JSON.stringify(auto, null, " ");
```

2. SessionStorage

- The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session.
- The data is deleted when the user closes the specific browser tab.
- See the demofiles for each

2. WebSQL

- Enables developers to use an SQLite database within a browser to store data
- Users Relative Tables and SQL syntax
- Enables transactions and rollbacks

2. WebSQL

- Creating and opening a database:

```
var db = openDatabase('mydb', '1.0', 'my
first database', 2 *1024 * 1024);
```

- Creating a table and inserting contents

```
var db = openDatabase('mydb', '1.0', 'my first database', 2 *
1024 * 1024);
db.transaction(function (tx) {
  tx.executeSql('CREATE TABLE IF NOT EXISTS foo (id unique,
text)');
  tx.executeSql('INSERT INTO foo (id, text) VALUES (1,
"synergies")');
});
```

2. WebSQL

- Selecting data and parsing results
- See the demofile for more info

```
tx.executeSql('SELECT * FROM foo', [], function (tx, results)
{
  var len = results.rows.length, i;
  for (i = 0; i < len; i++) {
    alert(results.rows.item(i).text);
  }
});
```

IndexedDB

- IndexedDB is basically a simple flat-file database with hierarchical key/value persistence and basic indexing.
- PRO: If you're a NoSQL type of person, then this might fit the bill perfectly.
- CON: Not yet available in most new browsers.
- CON: If you wanted SQL, you're not getting it here. Though in the future, it might be a great building block for implementing a SQL engine for the browser.

3. Working Offline with Appcache

- HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.
- Application cache gives an application three advantages:
- Offline browsing - users can use the application when they're offline
- Speed - cached resources load faster
- Reduced server load - the browser will only download updated/changed resources from the server

3. Working Offline with Appcache

- To enable application cache, include the manifest attribute in the document's <html>
- tag

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

- Every page with the manifest attribute specified will be cached when the user visits it.

3. Manifest file

- The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).
- The manifest file has three sections:
- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

3. Manifest file

- The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).
- The manifest file has three sections:
- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

3. Manifest file

- Example manifest file
- Note the comment line starting with #
- Cache is refreshed only if manifest file changes
- Comments can be used to change the file

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html
```

Questions or comments?

JavaScript and REST API's

Mika Stenberg

AJAX

- With AJAX it is easy to fetch data and integrate it as part of your webpage or web application
- You have already tried this with some simple XML files or URL's

What is a REST API

- **RESTful** web services are one way of providing interoperability between computer systems on the Internet.
- In a REST web service, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON or some other defined format.
- REST provides a way to make dynamic queries to a remote server using HTTP – no need to use Special database libraries
- Many databases (especially NoSQL db's) offer REST API's

Why a REST API

- Tens of Thousands of Public API's offer free data for the developers to build on
- Datasets include timetables, product catalogs, weather and statistical data, city / country demographic data, geographic data / maps etc.
- Opening the data enables new innovations and new ways to use it
  - https://www.publicapis.com/
  - http://www.programmableweb.com/apis/directory
  - http://data.europa.eu/euodp/en/data
  - http://apisuomi.fi/
  - https://www.avoindata.fi/fi

Example data sets available

- Finnish weather/air quality data:
  https://www.biomi.org/web/ilmanlaaturajapinta/
- VR train traffic info on realtime:
  https://rata.digitraffic.fi/
- Finnkino Movie Data:
  http://www.finnkino.fi/xml
- Spotify Music Data:
  https://developer.spotify.com/web-api/
- Alko Product Data API:
  http://www.alko.fi/api/product/Availability?productId=000706&cityId=tampere&language=fi

Example Use

- Request (parameters in bold):
  http://api.openweathermap.org/data/2.5/weather?**q=Helsinki**
  &**units=metric**&**mode=XML**&**APPID**=ff64c247a136f706923d1ee0d55d71e2

- Make note of the URL and the parameters after it are separated with & -sign
  - url: http://api.openweathermap.org/data/2.5/weather
  - ?q = City we are looking for
    &units= Metric / Imperial
    &mode = XML or JSON
    &APPID = developer key to prevent malicious activity

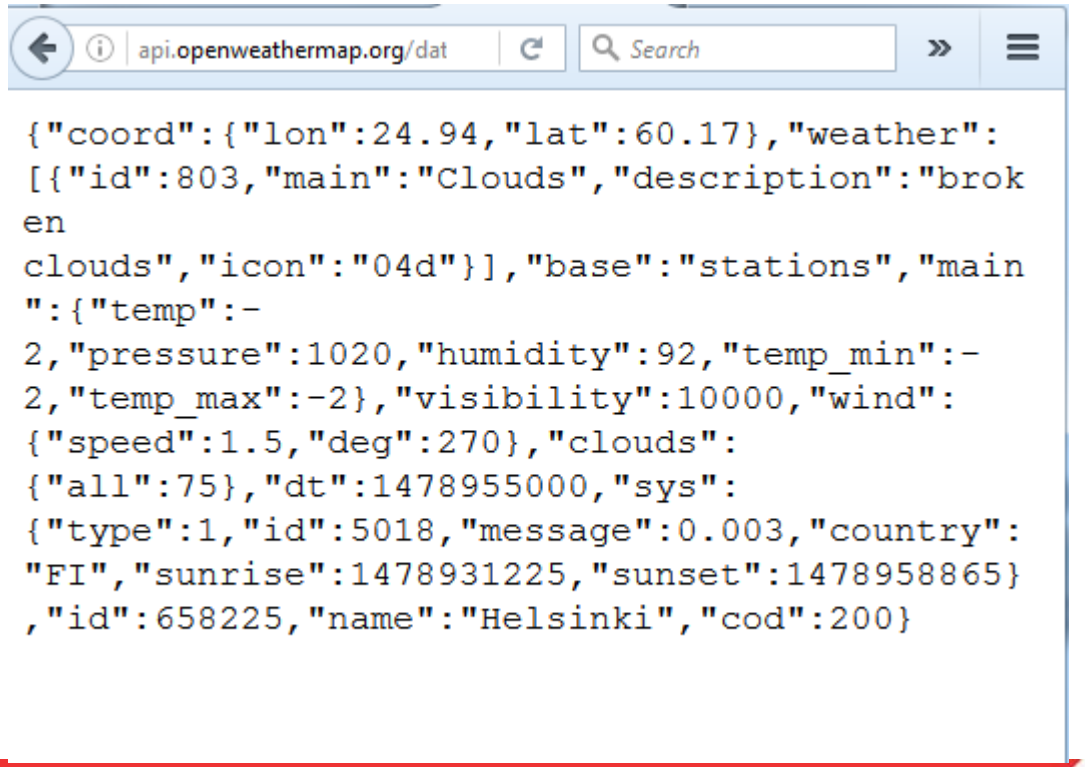
Example Use

- Request (parameters in bold):
  http://api.openweathermap.org/data/2.5/weather?**q=Helsinki**
  &**units=metric**&**mode=XML**&**APPID**=ff64c247a136f706923d1ee0d55d71e2
- Response:

```
- <current>
  - <city id="658225" name="Helsinki">
      <coord lon="24.94" lat="60.17"/>
      <country>FI</country>
      <sun rise="2016-11-12T06:13:41" set="2016-11-12T13:54:29"/>
    </city>
    <temperature value="-2.49" min="-3" max="-2" unit="metric"/>
    <humidity value="100" unit="%"/>
    <pressure value="1020" unit="hPa"/>
  - <wind>
      <speed value="6.18" name="Moderate breeze"/>
      <gusts/>
      <direction value="325.505" code="NW" name="Northwest"/>
    </wind>
    <clouds value="75" name="broken clouds"/>
    <visibility value="10000"/>
    <precipitation mode="no"/>
    <weather number="803" value="broken clouds" icon="04d"/>
    <lastupdate value="2016-11-12T11:50:00"/>
  </current>
```

Example Use

- Request (parameters in bold):
  http://api.openweathermap.org/data/2.5/weather?**q=Helsinki**
  &**units=metric**&**mode=JSON**&**APPID**=ff64c247a136f706923d1ee0d55d71e2
- Response in JSON:



JSON and XML compared

- Simple dataset both in XML and in JSON
- The following example defines an employees object, with an array of 3 employee records

```xml
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

```json
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Why JSON?

- For AJAX applications, JSON is faster and easier than XML

- Using XML
  - Fetch an XML document
  - Use the XML DOM to loop through the document
  - Extract values and store in variables

- Using JSON
  - Fetch a JSON string
  - JSON.Parse the JSON string

JSON = JavaScript Object

- JSON looks messy, but can be formatted nicely:
  - Ex. http://www.jsoneditoronline.org/
- Up to 10x faster to parse than XML
- Much smaller in size
- Easier to integrate into code – no parsing required


```
{
    "coord": {
        "lon": 24.94,
        "lat": 60.17
    },
    "weather": [
        {
            "id": 803,
            "main": "Clouds",
            "description": "broken clouds",
            "icon": "04d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": -2,
        "pressure": 1020,
        "humidity": 92,
        "temp_min": -2,
        "temp_max": -2
    },
    "visibility": 10000,
    "wind": {
        "speed": 1.5,
        "deg": 270
    },
    "clouds": {
        "all": 75
    },
    "dt": 1478955000,
    "sys": {
        "type": 1,
        "id": 5018,
        "message": 0.003,
        "country": "FI",
        "sunrise": 1478931225,
        "sunset": 1478958866
    },
    "id": 658225,
    "name": "Helsinki",
    "cod": 200
}
```

Parsing JSON data

- The text formatted data is placed in a variable

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

- Text data must be parsed to JSON

```
var obj = JSON.parse(text);
```

- Variable is used in the code as an array or an object, depending on the structure

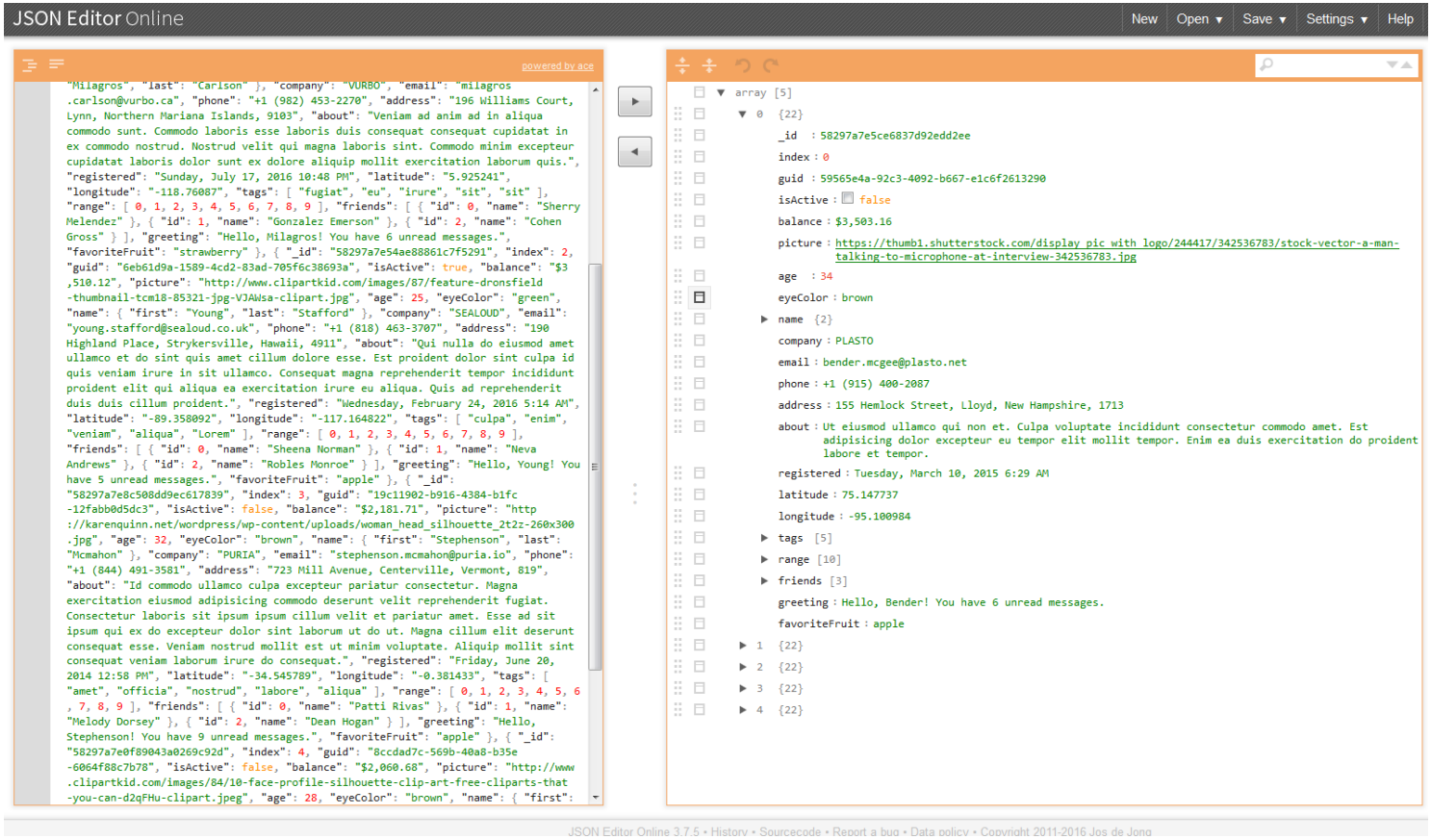
obj.employees[1].firstName
obj.employees[1].lastName;

More on this at:

- http://www.w3schools.com/js/js_json_intro.asp

Understanding the data is they key

- Use editors to understand the Structure, then dig out the [data]



Reading JSON data from web (REST)

- JSON works with AJAX in a similar fashion as XML

```
var url = "http://iceberg-cycle.codio.io/Project%202:%20Weather%20App/sampledata.js";
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", url, true);
xmlhttp.send();

xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        document.getElementById("content").innerHTML = xmlhttp.responseText;
        // Receive the response as JSON and parse it into a JS variable
        jsonObj = JSON.parse(xmlhttp.responseText);
        // We can now use it as any JS variable
        console.log ( jsonObj[0].name.first );
        console.log ( jsonObj[3].age );
    }
}
```

Reading JSON data from web (REST)

- JSON works with AJAX in a similar fashion as XML

```
var url = "http://iceberg-cycle.codio.io/Project%202:%20Weather%20App/sampledata.js";
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", url, true);
xmlhttp.send();

xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        document.getElementById("content").innerHTML = xmlhttp.responseText;
        // Receive the response as JSON and parse it into a JS variable
        jsonObj = JSON.parse(xmlhttp.responseText);
        // We can now use it as any JS variable
        console.log ( jsonObj[0].name.first );
        console.log ( jsonObj[3].age );
    }
}
```

The difference with XML is the use of *JSON.parse* – function.

And the fact that we can just start using the data as such!

Parsing the data on webpage

- We can select the data we want and output it to HTML

```
// JSON data is stored in data variable
var data = jsonObj;

// Create a loop, which will run through the JSON data array. All the data will be collected to out -variable
var out = "<table>";

for (var i=0; i < data.length; i++){
// for each loop round, we will create a new table for <tr> -tag and append (+=) the data to existing out -variable
    out += '<tr>';
    // For each cell, we will output data fields from JSON
    out += '<td>' + data[i].name.first + '</td>';
    out += '<td>' + data[i].name.last + '</td>';
    out += '<td>' + data[i].eyeColor+ '</td>';
    out += '<td>' + data[i].balance + '</td>';
    //out += '<td>' + data[i].picture + '</td>';
    out += '<td><img src="'+ data[i].picture+'"></td>';
    out += '</tr>';
}
// After all the data has been set, we will output closing tag for the table
out+="</table>";

// Place the newly created table in tabledata-div
document.getElementById("tabledata").innerHTML = out;
```

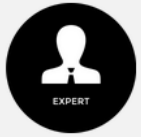
Loop will run through the data array and append the data to ”out” -variable

Finally we will just inject the data to the page

Workshop

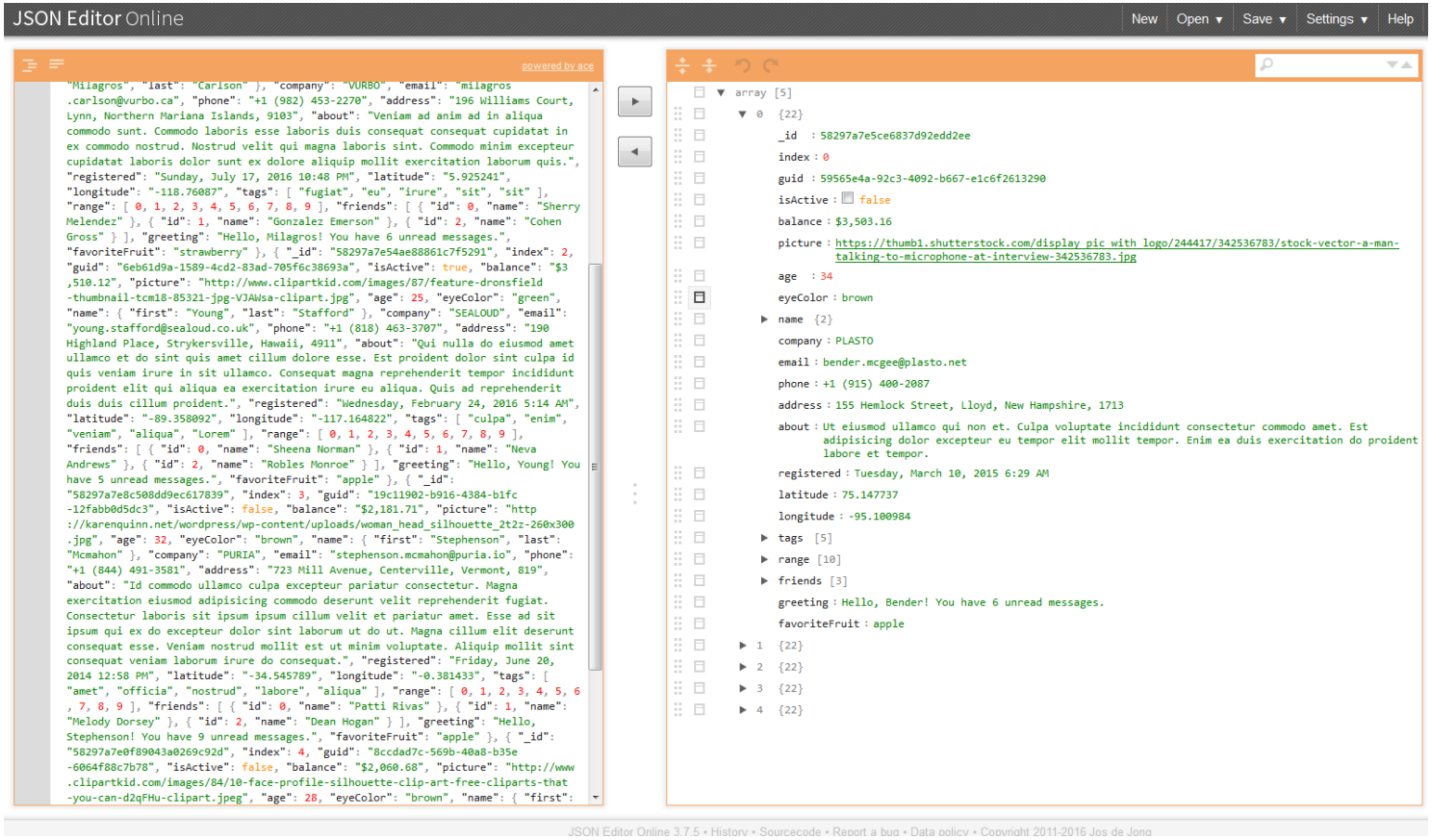
- The sample output of the data below

Parsed Data

Bender	Mcgee	brown	$3,503.16	
Milagros	Carlson	brown	$2,348.12	
Young	Stafford	green	$3,510.12	
Stephenson	Mcmahon	brown	$2,181.71	
Lakeisha	Merritt	brown	$2,060.68	

Understanding the data is they key

- Use editors to understand the Structure, then dig out the [data]



jQuery

Mika Stenberg

JavaScript

What will be discussed

1. What is jQuery
2. Setting up jQuery
3. DOM Scripting using jQuery
4. jQuery effects & animations
5. jQuery for Mobile?

1. What is jQuery

- You don’t have to write all the code yourself, there are libraries available, which one can utilize
- In JavaScript the most popular framework is jQuery
- jQuery simplifies coding;
  - querying elements and manipulating them much easier
  - has cool effects and animations which are easy to use
  - cross-browser and CSS3 –compliant
  - Is open source and has tons off addins and extras

2. Setting it up

- In order to use jQuery JavaScript library, we need to include it in our HTML –page
- There are two way to do this: 1) using CDN and 2) a local file

1) Reference the file using CDN (content delivery network):

```
<script
  src="https://code.jquery.com/jquery-3.7.1.min.js"
  integrity="sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="
  crossorigin="anonymous"></script>
```

2) Or use a local copy of the file and reference to it

```
<script src=" jquery-3.7.1.min.js "></script>
```

- And you’re done!

3. Selectors in jQuery

- jQuery looks a bit different, most of the lines begin with a $ - sign

Selecting Elements by ID

```
$( "#myId" ); // Note IDs must be unique per page.
```

Selecting Elements by Class Name

```
// The .html() method sets all the h1 elements' html to be "hello world":
$( "h1" ).html( "hello world" );
```

Selecting Elements by Attribute

```
$( "input[name='first_name']" ); // Beware, this can be very slow in older browsers
```

Selecting Elements by Compound CSS Selector

```
$( "#content" ).find( "h3" ).eq( 2 ).html( "new text for the third h3!" );
```

3. Selectors in jQuery

- jQuery offers pseudo-selectors for more complicated queries

Pseudo-Selectors

```
1  $( "a.external:first" );
2  $( "tr:odd" );
3
4  // Select all input-like elements in a form (more on this below).
5  $( "#myForm :input" );
6  $( "div:visible" );
7
8  // All except the first three divs.
9  $( "div:gt(2)" );
10
11 // All currently animated divs.
12 $( "div:animated" );
```

3. Editing content

- Editing the contents of an element

```
// The .html() method sets all the h1 elements' html to be "hello world":
$( "h1" ).html( "hello world" );
```

- Adding classes

```
// Attempting to call a jQuery method after calling a getter.
// This will NOT work:
$( "h1" ).html().addClass( "test" );
```

```
$( "#content" ).find( "h3" ).eq( 2 ).html( "new text for the third h3!" );
```

3. New Elements in jQuery

- Creating new elements

```
1 | // Creating new elements from an HTML string.
2 | $( "<p>This is a new paragraph</p>" );
3 | $( "<li class=\"new\">new list item</li>" );
```

- Adding newly created elements in the DOM

```
1 | // Getting a new element on to the page.
2 |
3 | var myNewElement = $( "<p>New element</p>" );
4 |
5 | myNewElement.appendTo( "#content" );
6 |
7 | myNewElement.insertAfter( "ul:last" ); // This will remove the p from #content!
8 |
9 | $( "ul" ).last().after( myNewElement.clone() ); // Clone the p so now we have two.
```

3. Events in jQuery

- Adding events

```
1  // Event setup using a convenience method
2  $( "p" ).click(function() {
3      console.log( "You clicked a paragraph!" );
4  });
```

- Determining which elements was clicked

```
1  $( "ul" ).on( "click", "li", function() {
2      console.log( "Something in a <ul> was clicked, and we detected that it was an <li> el
3  });
```

AJAX

- AJAX calls are shorter to write with jQuery as well

```
// Load demo_test.txt into div with an id #div1
$("#div1").load("demo_test.txt");

// Run Ajax call when the button is clicked
$("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

4. Effects & Animation

- For a complete reference, see documentation
- Some neat examples:

```
// Fade out an element
$('#logo').fadeOut(); // fadeIn();

// Completely hide an element
$('h1').hide(); // show();

// Animate slideup
$(ul).slideUp(); // slideDown();

// Perform custom animations
$('#content').animate();
```

5. Other jQuery projects

- jQuery for Mobile:
  Query Mobile is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices.

- jQuery UI
  Query UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

Learning resources

- jQuery has an extensive interactive learning platform with videos and slides available
- You can try it out at: try.jquery.com
- Complete jQuery documentation can be found at: api.jquery.com

Questions or comments?

Front End Frameworks

Mika Stenberg

Introductory task

- Answer the following questions:

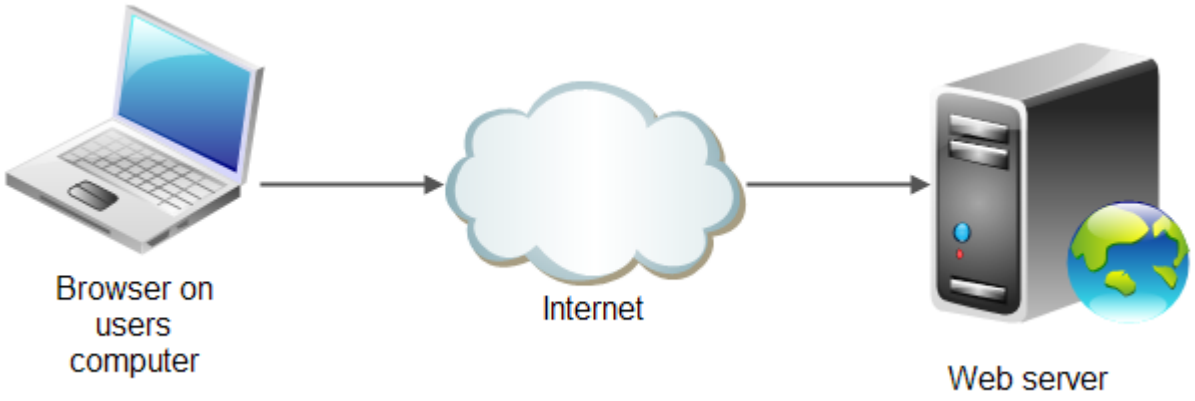
1. What are Front End Frameworks? Mention a few.
2. Why are they being used?
3. What do they offer to the developer?
4. How do they work?
5. How do they differ from JavaScript frameworks

- Use the following articles as a starting point and a reference:

1. http://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html
2. http://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/

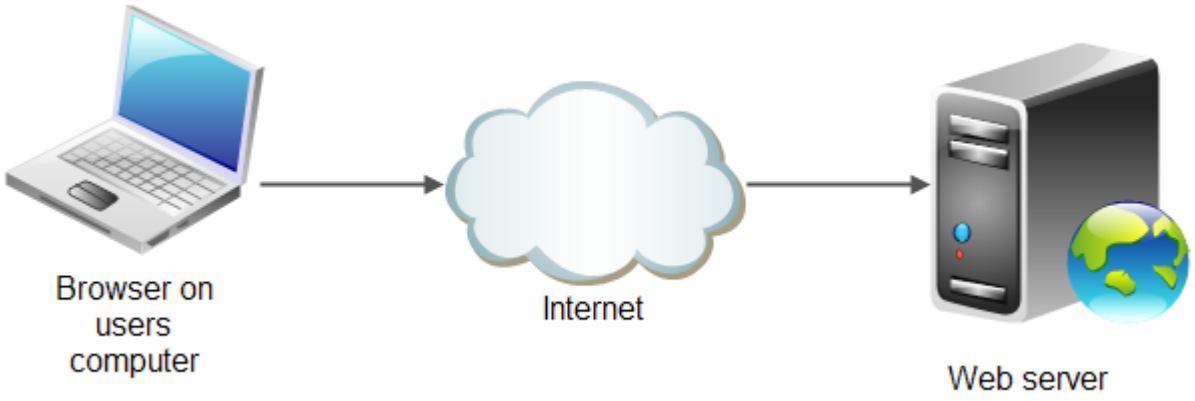
Web Development Overview

The World Wide Web



Web Development Overview

Front End Development **Back End Development**



Web Development Overview

Front End Development

PRESENTATION LAYER / FRONT-END

User interfaces

Languages: HTML, CSS, Javascript...

Frameworks: Bootstrap, Foundation 3, Grids Systems...

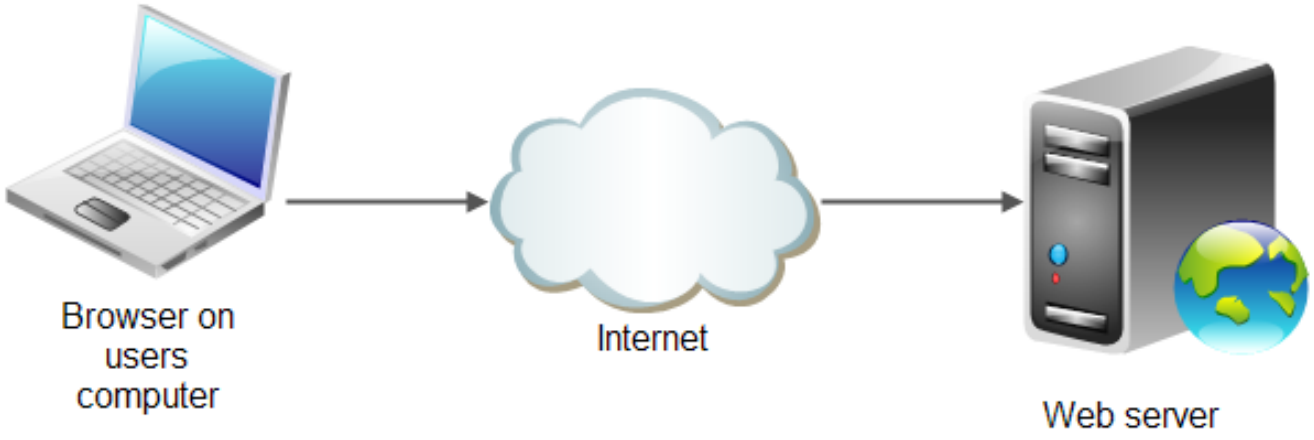
Back End Development

APPLICATION LAYER / BACK-END

Logic and operation of the website

Languages: PHP, PYTHON, RUBY, JAVA...

Frameworks: Symfony, Django, Ruby On Rails, Spring...



Web Development Overview

Front End Development Back End Development

PRESENTATION LAYER / FRONT-END

User interfaces

Languages: HTML, CSS, Javascript...
Frameworks: Bootstrap, Foundation 3, Grids Systems...

APPLICATION LAYER / BACK-END

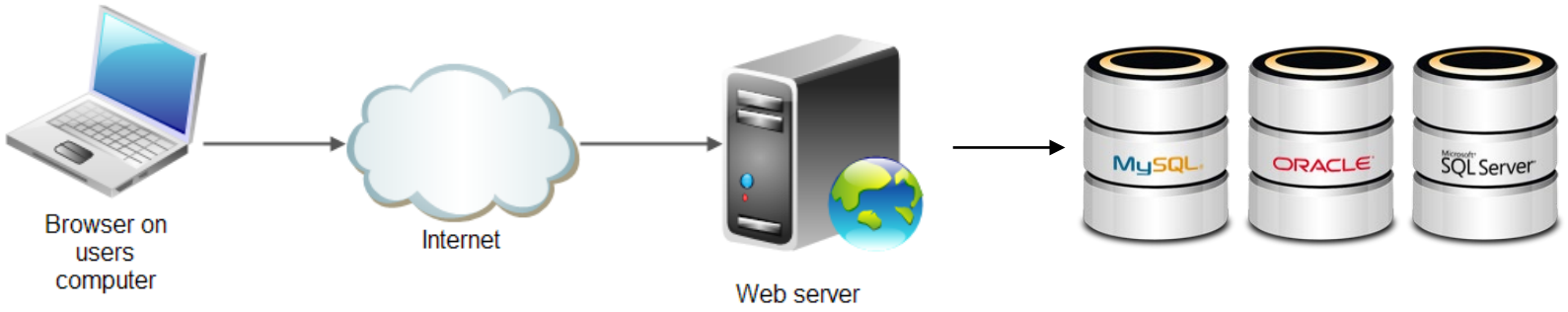
Logic and operation of the website

Languages: PHP, PYTHON, RUBY, JAVA...
Frameworks: Symfony, Django, Ruby On Rails, Spring...

DATA LAYER

Data persistence on databases

Technologies: MySQL, PostgreSQL, NoSQL...



Web Development Overview

Front End Development

Back End Development

Web Development Overview

Front End Development

Back End Development

LAMP - stack

MEAN - stack

What will be discussed

1. What are they?
2. How do they work?
3. Basics of Bootstrap
4. Final Project

1. What is it

- **CSS frameworks** are pre-prepared software frameworks that are meant to allow for easier, more standards-compliant web design using the Cascading Style Sheets language.
- Mostly design oriented and unobtrusive. This differentiates these from functional and full JavaScript frameworks.
- Most of these frameworks contain at least a grid.
- More functional frameworks also come with more features and additional JavaScript based functions

2. How does it work?

- CSS frameworks offer different modules and tools:
- reset style sheet
- grid especially for responsive web design
- web typography
- set of icons in sprites or icon fonts
- styling for tooltips, buttons, elements of forms
- parts of graphical user interfaces like accordion, tabs, slideshow or modal windows (Lightbox)
- equalizer to create equal height content
- often used css helper classes (*left*, *hide*)

3. List of CSS frameworks

Baseguide
Bootstrap
BoxySheets
Cascade Framework
Cascade Framework Light
Concise
floatz
Foundation
Ink
Inuitcss
Kathamo
Kube
Kule Lazy
Materialize
Modest Grid
Pure CSS
Responsee
Responsive BP
Responsive Grid System
Schema UI / Built with LESS
Semantic UI
Tacit
uikit
Unsemantic
Wee
Material Design Lite

4. Basics of Bootsrap

- Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

- UI elements: http://getbootstrap.com/examples/theme/

Basic of Bootstrap grid

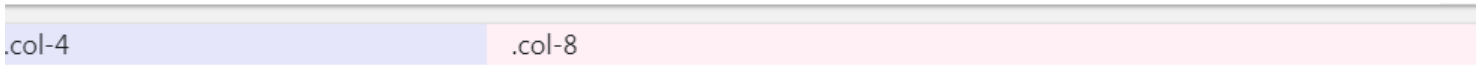
Bootstrap Grid System

Bootstrap's grid system allows up to 12 columns across the page.

If you do not want to use all 12 column individually, you can group the columns together to create wider columns:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Basic of Bootstrap grid




```
<div class="row">
  <div class="col-4" style="background-color:lavender;">.col-4</div>
  <div class="col-8" style="background-color:lavenderblush;">.col-8</div>
</div>
```

Basic of Bootstrap responsive grid

Three Equal Columns

.col-sm-4	.col-sm-4	.col-sm-4

The following example shows how to get a three equal-width columns starting at tablets and scaling to large desktops. On mobile phones, the columns will automatically stack:

Example

```
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
</div>
```

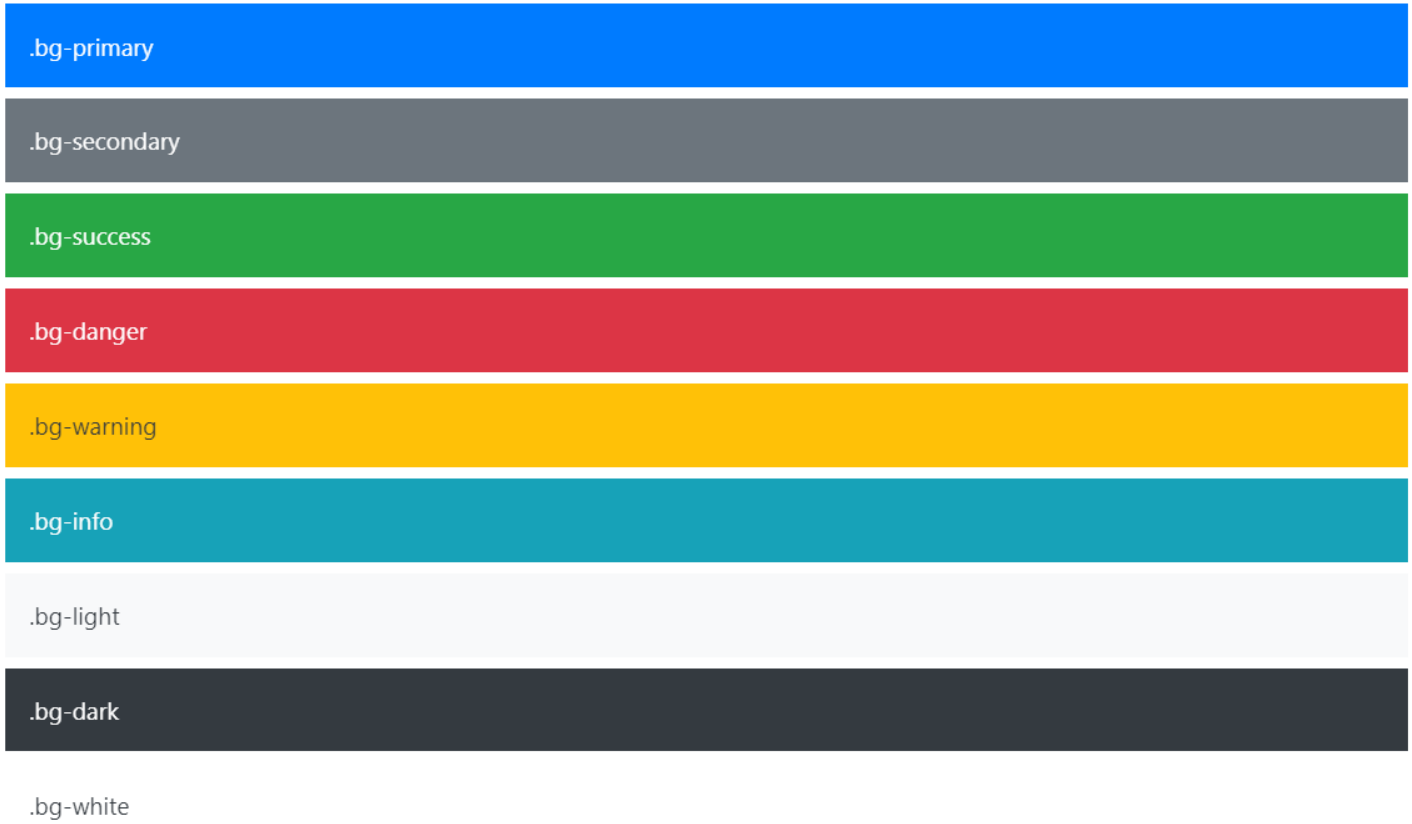
Grid Classes

The Bootstrap grid system has four classes:

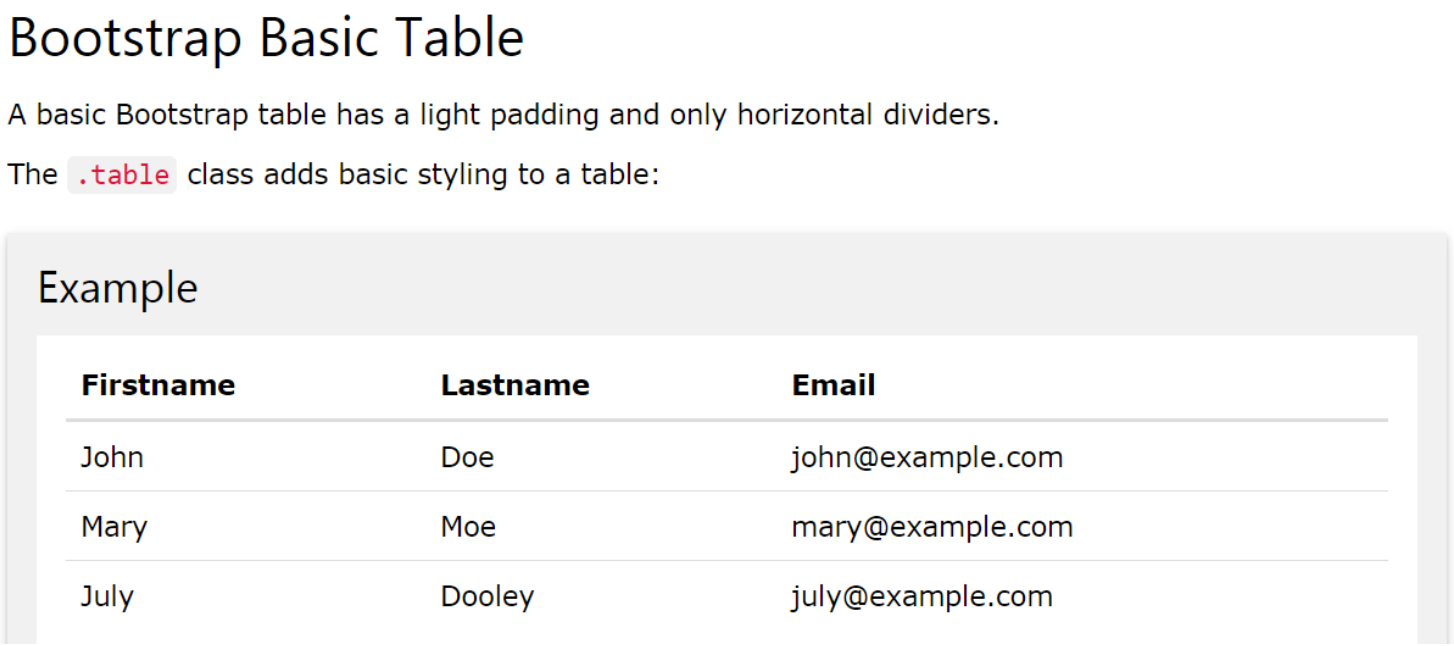
- `xs` (for phones - screens less than 768px wide)
- `sm` (for tablets - screens equal to or greater than 768px wide)
- `md` (for small laptops - screens equal to or greater than 992px wide)
- `lg` (for laptops and desktops - screens equal to or greater than 1200px wide)

The classes above can be combined to create more dynamic and flexible layouts.

Default Colorsets



Basic of Bootstrap table



Basic of Bootstrap table

Striped Rows

The `.table-striped` class adds zebra-stripes to a table:

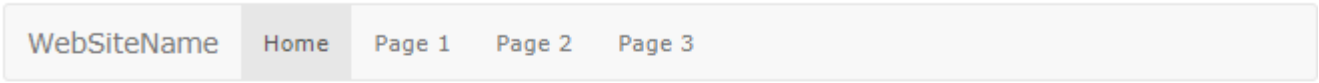
Example

Firstname	Lastname	Email
John	Doe	john@example.com
Mary	Moe	mary@example.com
July	Dooley	july@example.com

Basic of Bootstrap Navigation

Navigation Bars

A navigation bar is a navigation header that is placed at the top of the page:



With Bootstrap, a navigation bar can extend or collapse, depending on the screen size.

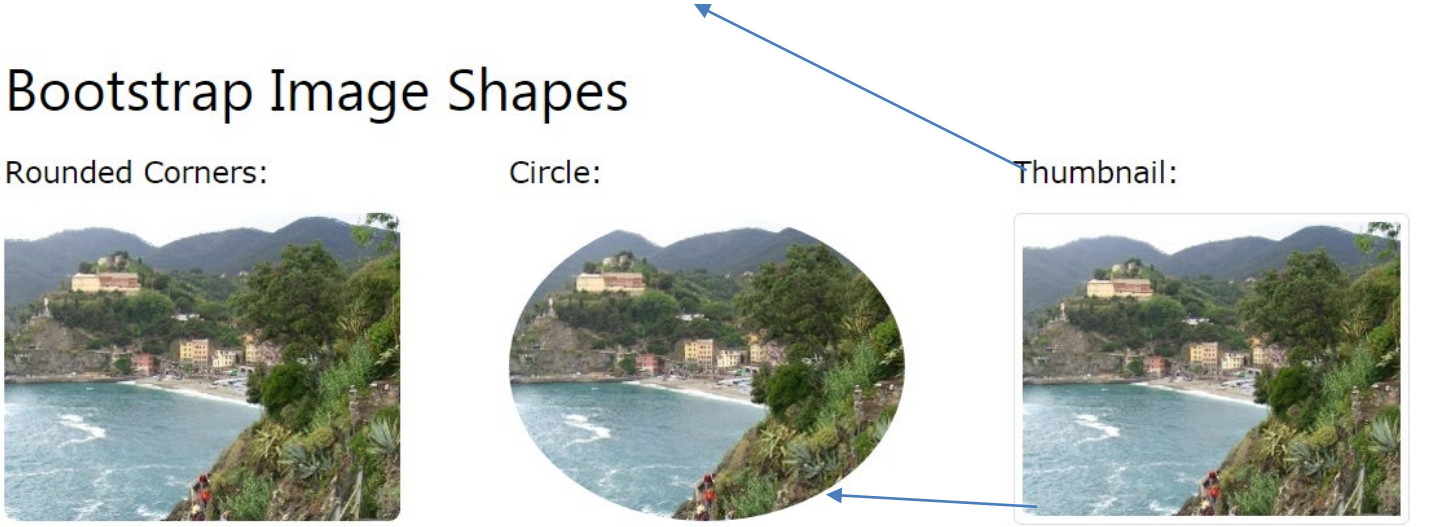
A standard navigation bar is created with `<nav class="navbar navbar-default">`.

The following example shows how to add a navigation bar to the top of the page:

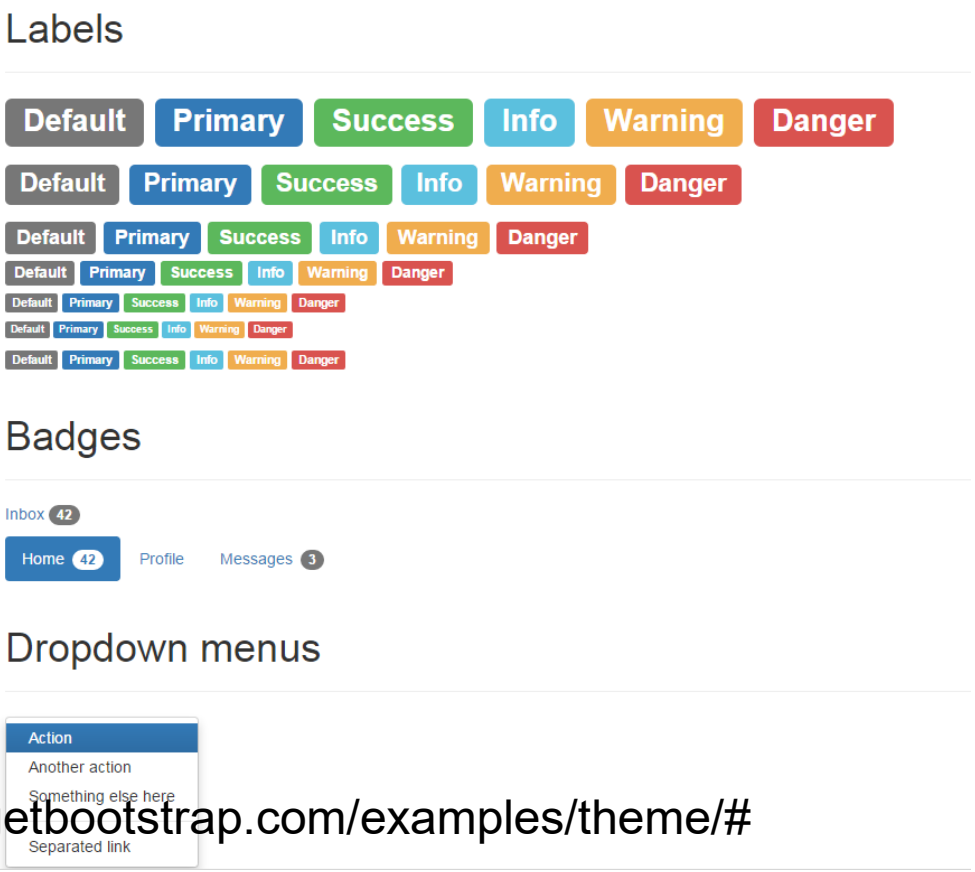
Example

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#">Page 1</a></li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </div>
</nav>
...
```

Basic of Bootstrap images

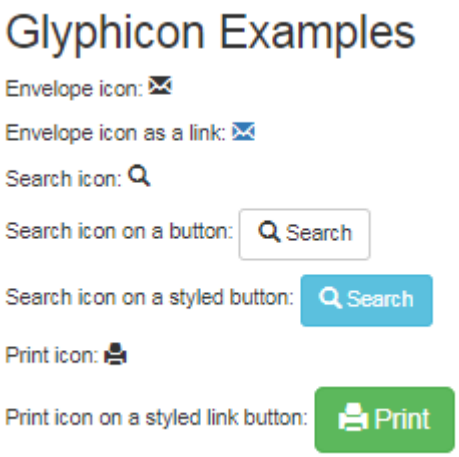


Basic of Bootstrap images



http://getbootstrap.com/examples/theme/#

Basic of Bootstrap images



Bootstrap Glyph Example

```
<p>Envelope icon: <span class="glyphicon glyphicon-envelope"></span></p>
<p>Search icon: <span class="glyphicon glyphicon-search"></span></p>
<p>Print icon: <span class="glyphicon glyphicon-print"></span></p>
```

```
<p>Print icon: <span class="glyphicon glyphicon-print"></span></p>
<p>Print icon on a styled link button:
  <a href="#" class="btn btn-success btn-lg">
    <span class="glyphicon glyphicon-print"></span> Print
  </a>
</p>
```

https://glyphicons.bootstrapcheatsheets.com/

Case Study: Examine the demo page

- Try to reason how it works and which parts do what. Browse the files in file explorer and view them in browser. Study the source code with (Ctrl+U) and developer tools (F12).

1. **Responsive Design:** Open the index.html page in browser. Resize browser window and see how the page responds. Then Open bootsrap.css and look for **@media** -tags. What do they do and how do they work?

2. **Grid system:** What kind of grid is used on page? How are rows and columns divided? What classes are used on columns and why?

3. **Responsive images:** Images are responsive and resized when window is resized. How is this achieved, which classes are used and what properties are set in that class?

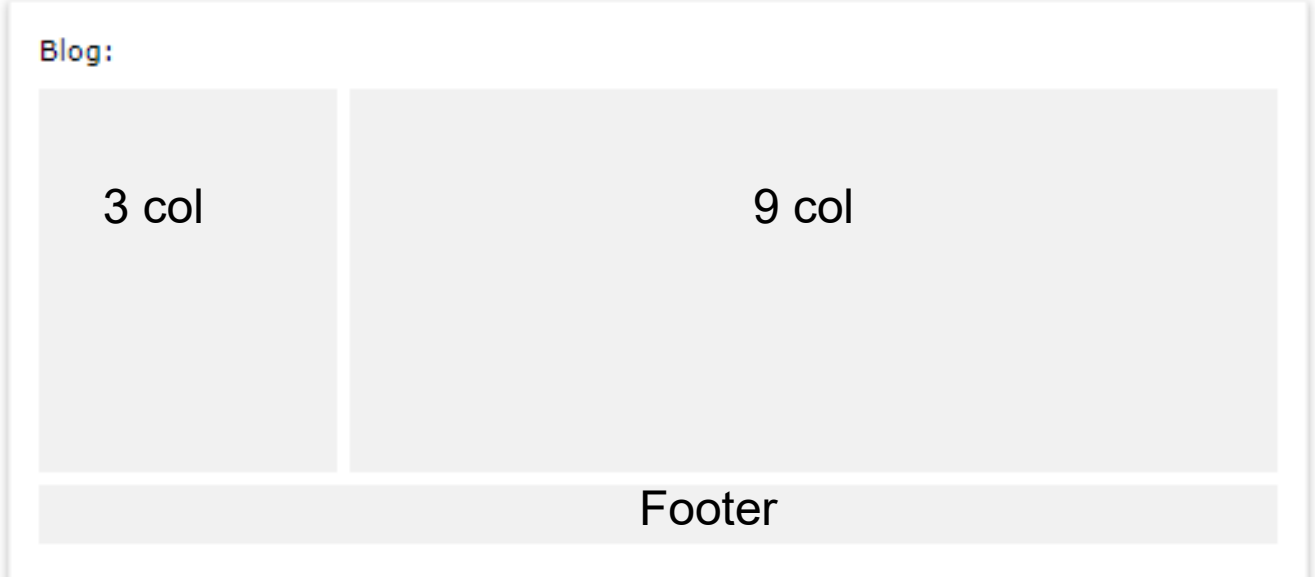
4. **UI Components:** There are some blue buttons visible on the page. How are they implemented? Use the Bootstap theme page as a reference.
   Change the style and look of the buttons by changing the classes used.

5. Add some UI components from Bootstrap theme to the page. Insert at least one **Alert -component** and one **Panel-component**. Pick one element of your choice from the list and add it to the page.

Questions or comments?

Exercises

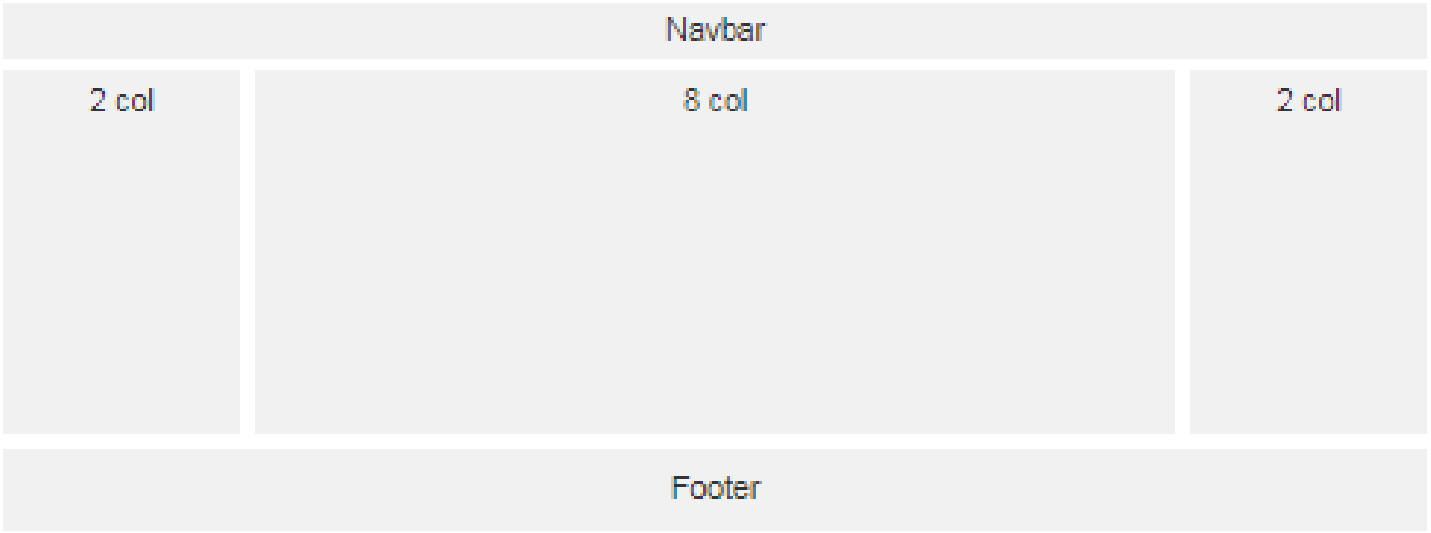
- Reproduce the following layout with Bootstrap



Exercises

- Reproduce the following layout with Bootstrap

Webpage:



Exercises

- Reproduce the following layout with Bootstrap

Project 3: Front End Frameworks

- Teams of max. 3 persons
- Your final project is to **use a Front End of Your Choice** to **implement a User Interface** for Project 1 or Project 2 implemented earlier in the course
- The page should implement **a responsive grid**, **custom UI elements** such as **buttons or panels/alerts, custom typography etc.**
- No JavaScript coding is necessary, only the HTML/CSS around the APP's is about to change
- Feel free to add some nice UI effects using jQuery or other

jQuery

Mika Stenberg

What will be discussed

1. What is jQuery
2. Setting up jQuery
3. DOM Scripting using jQuery
4. jQuery effects & animations
5. jQuery for Mobile?

1. What is jQuery

- You don’t have to write all the code yourself, there are libraries available, which one can utilize
- In JavaScript the most popular framework is jQuery
- jQuery simplifies coding;
  - querying elements and manipulating them much easier
  - has cool effects and animations which are easy to use
  - cross-browser and CSS3 –compliant
  - Is open source and has tons off addins and extras

2. Setting it up

- In order to use jQuery JavaScript library, we need to include it in our HTML – page
- There are two way to do this: 1) using CDN and 2) a local file

1) Reference the file using CDN (content delivery network):

```
<script src="http://code.jquery.com/jquery-2.1.4.min.js
"></script>
```

2) Or use a local copy of the file and reference to it

```
<script src="jquery-2.1.4.min.js"></script>
```

- And you’re done!

3. Selectors in jQuery

- jQuery looks a bit different, most of the lines begin with a $ - sign

Selecting Elements by ID

```
$( "#myId" ); // Note IDs must be unique per page.
```

```
// The .html() method sets all the h1 elements' html to be "hello world":
$( "h1" ).html( "hello world" );
```

Selecting Elements by Class Name

```
$( ".myClass" );
```

Selecting Elements by Attribute

```
$( "input[name='first_name']" ); // Beware, this can be very slow in older browsers
```

```
$( "#content" ).find( "h3" ).eq( 2 ).html( "new text for the third h3!" );
```

Selecting Elements by Compound CSS Selector

```
$( "#contents ul.people li" );
```

3. Selectors in jQuery

- jQuery offers pseudo-selectors for more complicated queries

Pseudo-Selectors

```
1  $( "a.external:first" );
2  $( "tr:odd" );
3
4  // Select all input-like elements in a form (more on this below).
5  $( "#myForm :input" );
6  $( "div:visible" );
7
8  // All except the first three divs.
9  $( "div:gt(2)" );
10
11 // All currently animated divs.
12 $( "div:animated" );
```

3. Editing content

- Editing the contents of an element

```
1 | // The .html() method sets all the h1 elements' html to be "hello world":
2 | $( "h1" ).html( "hello world" );
```

- Adding classes

```
1 | // Attempting to call a jQuery method after calling a getter.
2 | // This will NOT work:
3 | $( "h1" ).html().addClass( "test" );
```

```
1 | $( "#content" ).find( "h3" ).eq( 2 ).html( "new text for the third h3!" );
```

3. New Elements in jQuery

- Creating new elements

```
1 | // Creating new elements from an HTML string.
2 | $( "<p>This is a new paragraph</p>" );
3 | $( "<li class=\"new\">new list item</li>" );
```

- Adding newly created elements in the DOM

```
1 | // Getting a new element on to the page.
2 |
3 | var myNewElement = $( "<p>New element</p>" );
4 |
5 | myNewElement.appendTo( "#content" );
6 |
7 | myNewElement.insertAfter( "ul:last" ); // This will remove the p from #content!
8 |
9 | $( "ul" ).last().after( myNewElement.clone() ); // Clone the p so now we have two.
```

3. Events in jQuery

- Adding events

```
1 | // Event setup using a convenience method
2 | $( "p" ).click(function() {
3 |     console.log( "You clicked a paragraph!" );
4 | });
```

- Determining which elements was clicked

```
1 | $( "ul" ).on( "click", "li", function() {
2 |     console.log( "Something in a <ul> was clicked, and we detected that it was an <li> e
3 | });
```

AJAX

- AJAX calls are shorter to write with jQuery as well

```
// Load demo_test.txt into div with an id #div1
$("#div1").load("demo_test.txt");
```

```
// Run Ajax call when the button is clicked
$("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

4. Effects & Animation

- For a complete reference, see documentation
- Some neat examples:

```
// Fade out an element
$('#logo').fadeOut(); // fadeIn();

// Completely hide an element
$('h1').hide(); // show();

// Animate slideup
$(ul).slideUp(); // slideDown();

// Perform custom animations
$('#content').animate();
```

5. Other jQuery projects

- jQuery for Mobile:
  Query Mobile is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices.

- jQuery UI
  Query UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

Learning resources

- jQuery has an extensive interactive learning platform with videos and slides available
- You can try it out at: try.jquery.com
- Complete jQuery documentation can be found at:
  api.jquery.com

Questions or comments?