JavaScript pt. 1: Intro and Ajax

Kianoosh Abbasi

CSC309 Winter 2022

Some content is from Dr. Sadia Sharmin's slides of CSC309 Winter 2021: www.rainsharmin.com

So far

- Front-end:
  - HTML: tags and forms
  - CSS: styles, selectors, layout
- Back-end: Django
  - Setup, simple views, forms, templates
  - MVC, database, ORM, Auth
  - Custom models, migrations, class-based views
  - JSON, Restful APIs

This session

- Intro to JavaScript
- DOM
- Getting and manipulating elements
- Asynchronous requests: Ajax

Front-end so far

- HTML: Describes what should be on our page
- CSS: Describes how elements should look like
- But the web age is not interactive!
  We need something that responds to events and user actions
- JavaScript: a language that browsers understand!

Netscape - [Version 2.02]

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location: about:

What's New! What's Cool! Handbook Net Search Net Directory Software

Netscape Navigator (TM)

Version 2.02

This software is subject to the license agreement set forth in the license. Please read and agree to all terms before using this software.

Report any problems through the feedback page.

NETSCAPE

JAVA COMPATIBLE

Contains Java™ software developed by Sun Microsystems, Inc.

RSA PUBLIC KEY CRYPTOSYSTEM

Contains security software from RSA Data Security, Inc.

This version supports International security with RSA Public Key Cryptograph

Any provision of Netscape Software to the U.S.Government is with "Restricted rights" as follows: Use, duplication or disclosure by the Government is subje

Java vs JavaScript

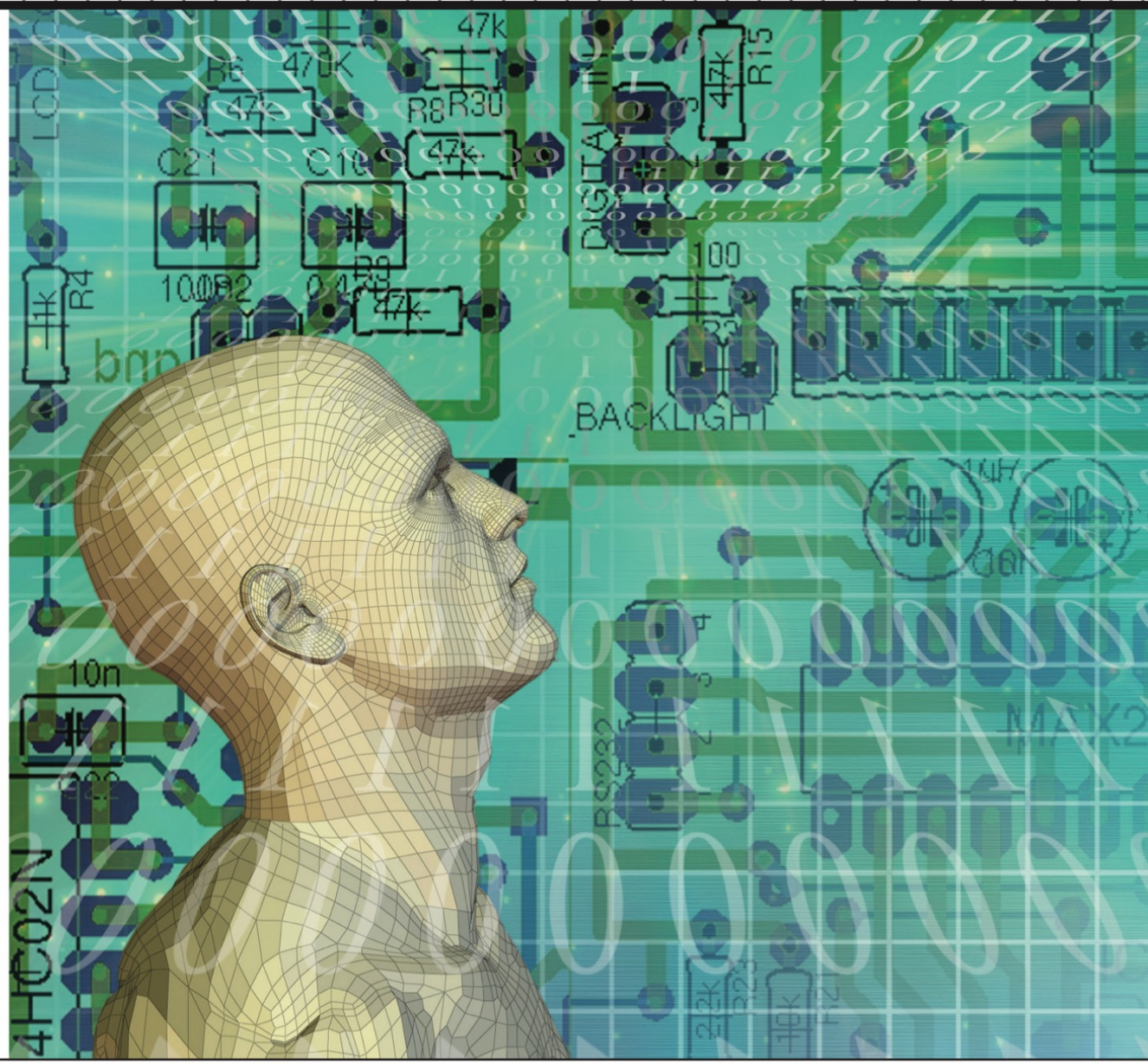
- Eich's script language had a somewhat similar syntax to Java

- Netscape-Sun deal:
  - Netscape browser will support Java apps
  - Eich's language will be called JavaScript!

- No further relevance between Java and JavaScript!

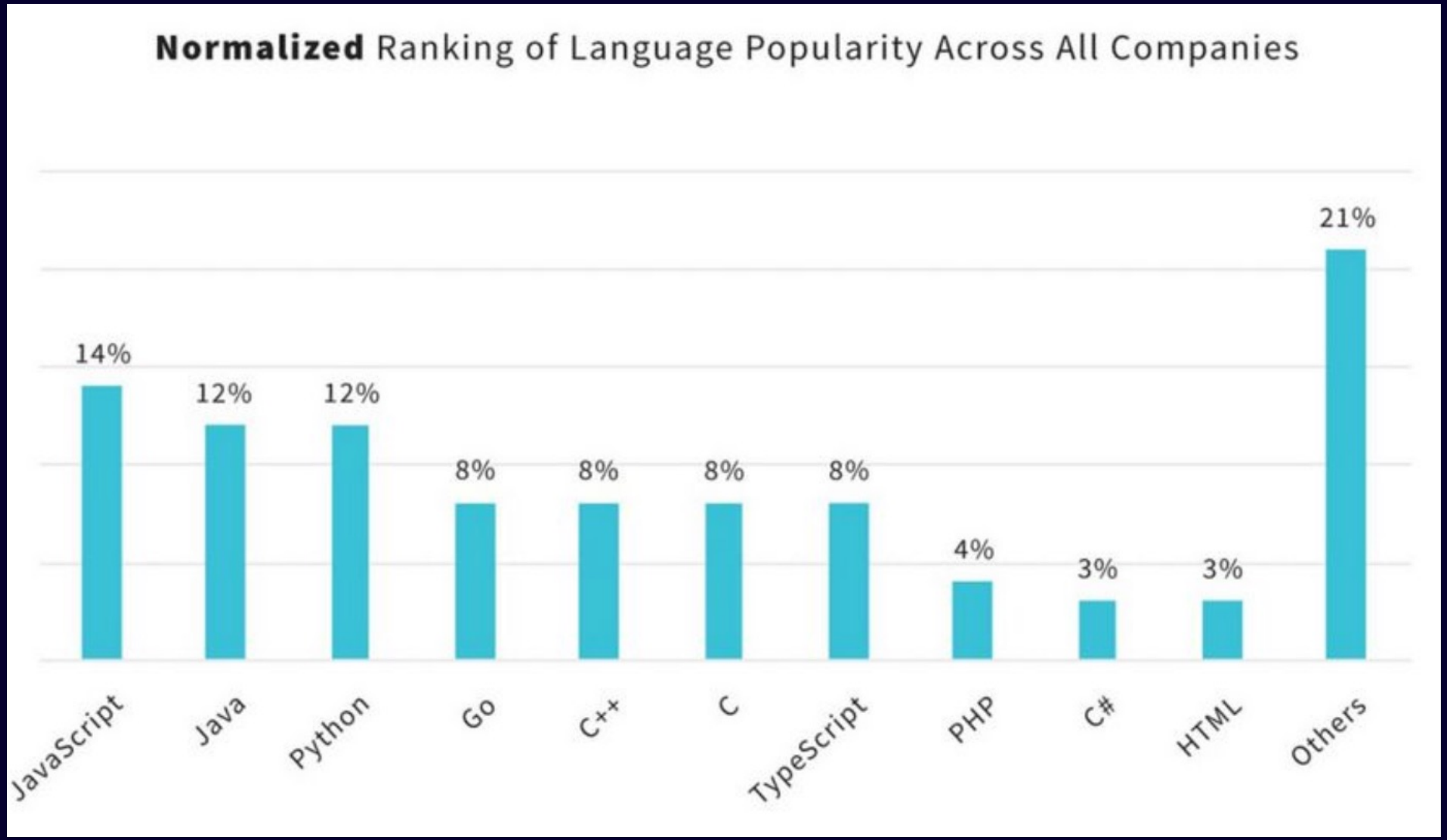
JavaScript: Designing a Language in 10 Days

Charles Severance
University of Michigan



There we go!

Visit: https://madnight.github.io/githut



Source: https://solutionshub.epam.com/blog/post/programming-language-popularity-on-github

- JavaScript is a scripting language
  - Interpreted at runtime
  - No JAR or exe file
- Almost all browsers have a JS interpreter
  - Run JS code accompanying the HTML (i.e., static files)
- Is it exclusive to client-side web applications?
- Answer: NO!
  - Example: NodeJS (more on that later)
  - Moreover, JS is just a language! You sort an array of integers with it

Syntax

- Declaring variables

```
var x = 5;
var y = "hello";
console.log(x + y);
```

- Data types:

  Number, string, boolean, undefined

  Object, function

- JS is dynamically-typed (like Python)

Objects

- Examples:

```
var cars = ["Saab", "Volvo", "BMW"];
var person = {firstName: "John", lastName: "Doe", age: 50,
eyeColor: "blue"};
var ref = null;
```

- Looks like JSON, doesn't it?
  That's where the JS in JSON comes from

- Note: null is different than undefined
  `typeof(null)` returns object, while `typeof(undefined)` returns undefined!

Properties

- Examples:

```
person.firstName = "Joe";
person["lastName"] = "Jordan";
cars[0] = 1;
cars.push(2323);
```

- Objects (including arrays) are the only mutable types in JS

Functions

- Syntax:

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

- Properties can be functions (methods)

```
var obj = {f: function(x) {
   return x + 2
}}

cars.clear = function(){
    this.length = 0;
}
```

Classes

Visit https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

- A template for creating objects
- Are in fact special functions

  Check `typeof(Rectangle)`
- Classes support inheritance

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  // Getter
  get area() {
    return this.calcArea();
  }
  // Method
  calcArea() {
    return this.height * this.width;
  }
}

const square = new Rectangle(10, 10);

console.log(square.area); // 100
```

Conditions

- If statements:

```
if (typeof(cars[0]) === "number" && cars[0] < 0)
    cars[0] *= -1;
else
    console.log("Bad element");
```

- Important: notice `===`

  Visit https://codeahoy.com/javascript/2019/10/12/==-vs-===-in-javascript/

More statements

- While loops:

```
while (cars.length > 0){
    cars.pop();
}
```

- Switch statement:

```
switch(cars[0]){
    case 1:
        console.log("int");
        break;
    case "name":
        console.log("str");
        break;
    case x:
        console.log("var " + x);
        break;
    default:
        console.log("none");
}
```

For loops

- Classic for loop:

```
for (var i=0; i<10; i++)
    console.log(i * i * i);
```

- Iterable objects:

```
for (name of names)
    console.log("There is a " + name)
```

- Array-specific forEach:

```
names.forEach(function(index, name){
    console.log(name + " at index " + index);
})
```

Scope

Visit: https://www.w3schools.com/js/js_scope.asp

- Three types of scope:
  - Global scope
  - Function scope
  - Block scope

- Global scope
  - Outside any function
  - Variables can be accessed from anywhere in the program

Function scope

- Variables defined anywhere inside a function are local to that function
- Can be used anywhere inside that function
- Cannot be used outside that function

```
// code here can NOT use carName

function myFunction() {
  var carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

Block scope

- To limit a variable to its block inside the function, use let

```
function f(n){
    if (n > 10){
        var tmp = 2;
    }
    // tmp CAN be accessed here
}
```

```
function f(n){
    if (n > 10){
        let tmp = 2;
    }
    // tmp can NOT be accessed here
}
```

Let vs var

- At global and function scopes, let and var work the same
- var supports redeclaration, while let does not
- Both support re-assignment. Use const to disallow it
- let is more like regular variables in other languages
  - Preferred over var

Manipulating the web page

```
alert("Are you REALLY sure you want to leave??")
```

Where to put JS

- JS code should be placed inside the <script> tag

- Inline JS:

```
<script>
   console.log(1 + 2 + 3)
</script>
```

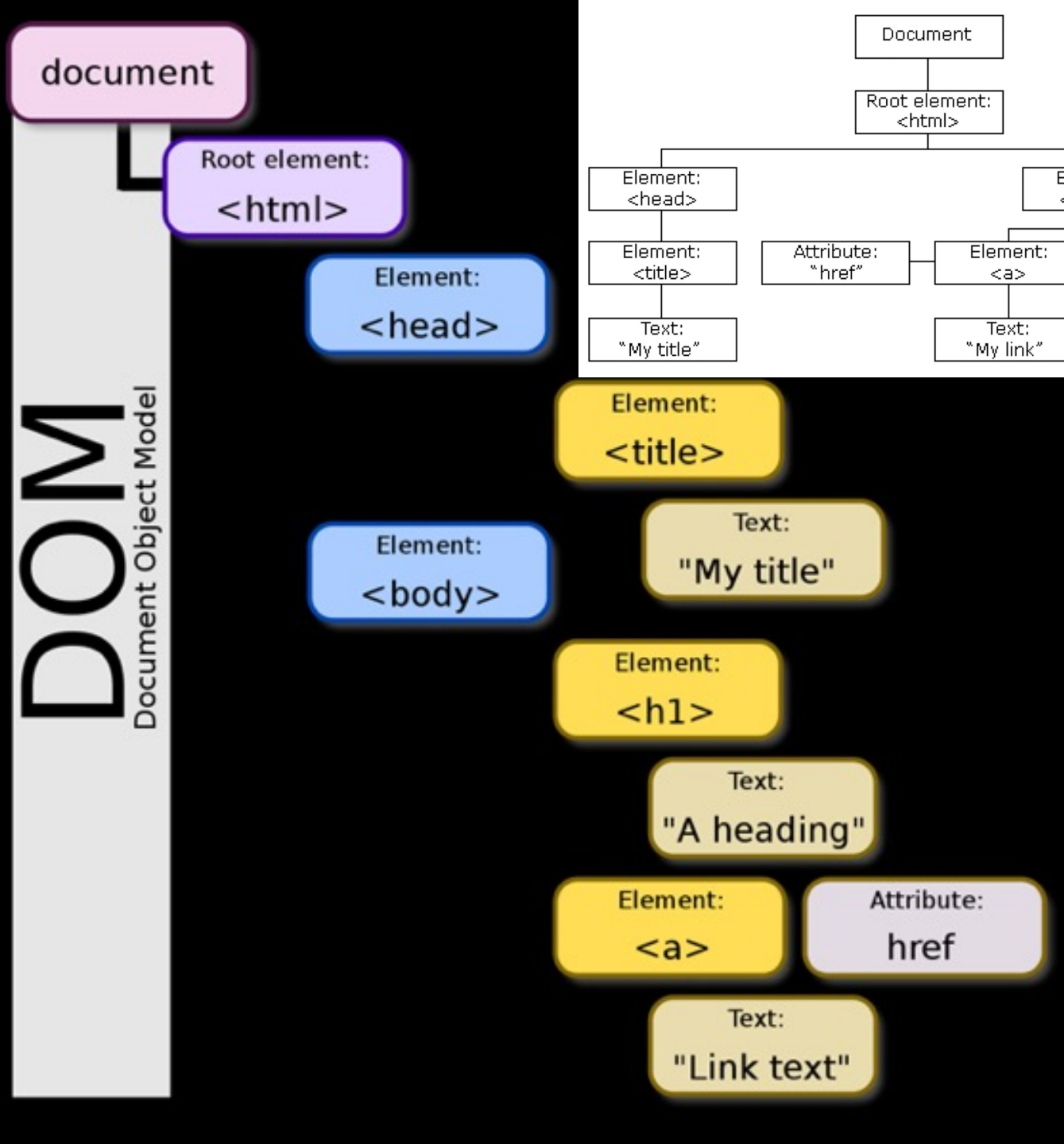
- JS file

```
<script src="city_region_dropdown.js"></script>
```

Document object model

- Browser creates the DOM tree of the page
- Each element is a node
- Child elements are children of the parent node
- Scripts access DOM through the document variable

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="...">Link
    text</a>
  </body>
</html>
```



Getting elements

- Various ways to get an element

```
document.getElementById("st-2")
document.getElementsByClassName("ne-share-buttons")
document.getElementsByTagName("ul")

document.querySelector("#submit-btn")
document.querySelectorAll(".col-md-12")
```

- Good exercise at:

  https://javascript.info/task/find-elements/table.html

Navigating through DOM

- Relevant nodes can be accessed through properties

  `parentNode, firstChild, lastChild, childNodes, nextSibling`

- Example

```
let img = document.querySelector(
                "body section:first-child > img")
let par = img.parentNode

console.log(par.childNodes.length);
```

Manipulating elements

- Element properties
  innerHTML, style, getAttribute()

- Example

```
let body = document.body
body.innerHTML = "<h3>hello!</h3>"

h3 = document.getElementsByTagName("h3")
h3.style.color = "green"
h3.setAttribute("class", "title")
console.log(h3.getAttribute("style"))
```

Events

Visit https://www.w3schools.com/tags/ref_eventattributes.asp

- Various events are monitored by the browser

- document events

  onload, onkeydown, onkeyup

- Element events

  onclick, onmouseover, ondrag, oncopy, onfocus, onselect, onsubmit

Events

- You can define a function

```
h3.onclick = function() {
    this.innerHTML = "you just clicked on me!"
}
```

- Alternative:

```
<script>
    function h3click(h3){
        h3.style.color = "blue"
    }
</script>

<h3 onclick="h3click(this)" onmouseover="console.log(new
Date())"></h3>
```

Exercise: A form with client-side validation

- Examples:

  Checks if a security question is answered correctly

  Checks if the email input is valid

  Checks password and repeat password are the same

- Errors should appear dynamically and disappear if user has fixed the issue

Asynchronous requests

Requests

- Currently, one main request is made to the server (upon entering the URL or submitting a form)
- Response is rendered and additional requests made by browser to fetch static data (js, css, images, fonts, etc.)
- This way entails a full reload for just one request!!

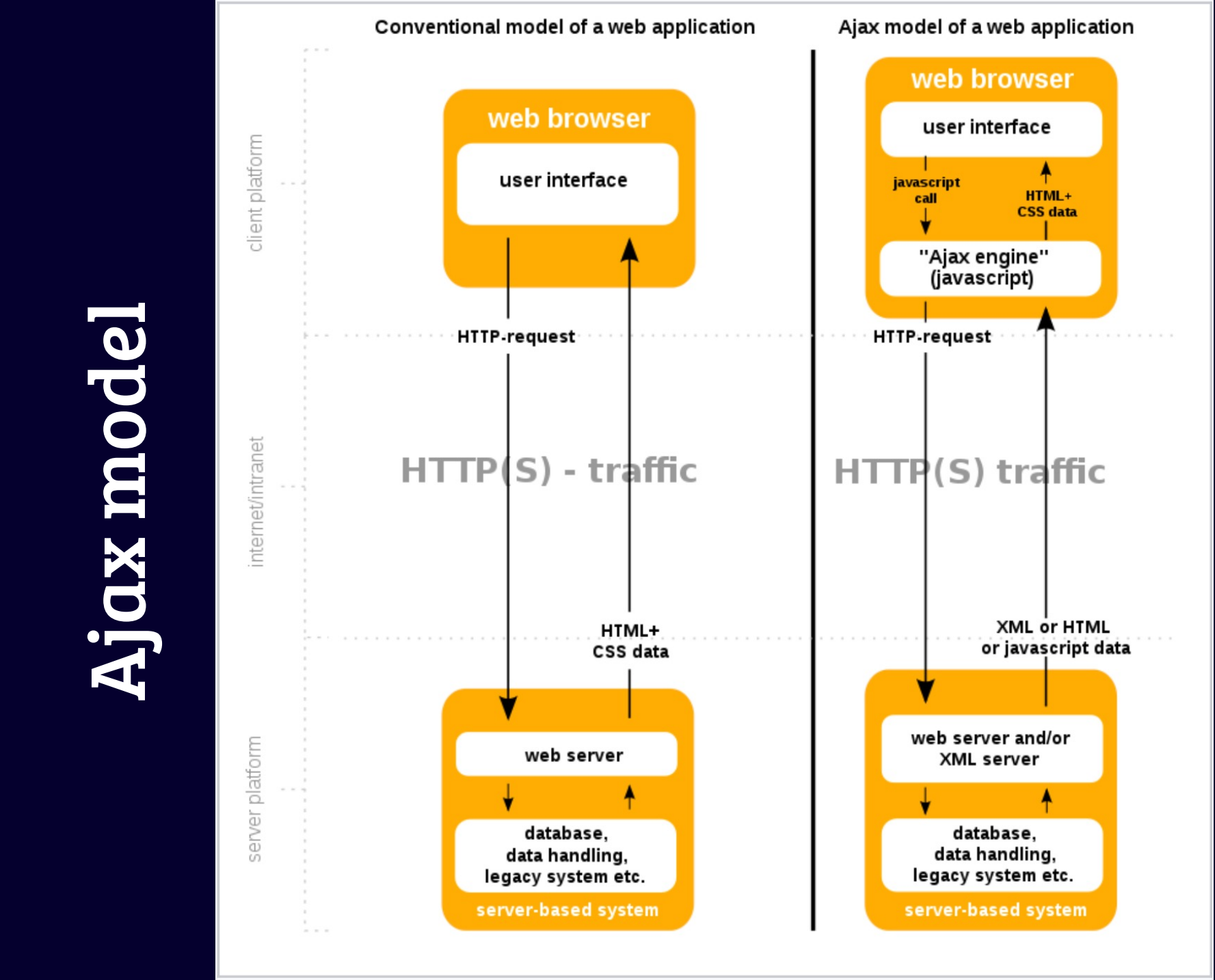
Solution

- Asynchronous JavaScript and XML (Ajax)

- Browser sends the request in background
  Does not block the main thread

- Response is handled by a series of events and callbacks
  Further changes are made to the document

Ajax model



Source: https://en.wikipedia.org/wiki/Ajax_(programming)

Why is it important

- Offers more control over the web page
  - You lose everything once the browser exits the current page!
- Most modern websites do not use the submit feature
  - Instead, they send an Ajax request and handle response
  - Client-side JS code redirects if necessary
- Basis for single-page frameworks like React

Sending an Ajax request

- Instantiate the request

```
let req = new XMLHttpRequest();
```

- Define a handler for onreadystatechange

```
req.onreadystatechange = function(){
    // Process the server response here.
};
```

- Set method & endpoint and send!

```
req.open("GET", "http://localhost/accounts/");
req.send();
```

Ajax Example

Visit https://www.w3schools.com/xml/ajax_intro.asp

```
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {
    if (this.readyState == XMLHttpRequest.DONE && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };

  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>
```

- You must always check if the ready state is DONE to access `responseText`
  - Other values: loading, loaded, interactive
- GET params are appended to the URL
- POST data is the body of HTTP request
  - Should be specified as the argument to `req.send()`
- Verbose, isn't it?
  - Everything with pure JS is verbose!
  - Next session, we will talk about a library that makes lives easier!

This session

- Intro to JavaScript
- DOM
- Getting and manipulating elements
- Asynchronous requests: Ajax

Next session

- Intro to jQuery

- Advanced JS (React prep)
  - Closure
  - Arrow functions
  - Promises

Final notes

- Project phase 2 due is next Friday
- Keep in mind what your front-end would look like
  Design your APIs based on that
- Very important to implement reasonable APIs
  Saves you a lot of time at phase 3 and front-end connection
- Sign up for mentor sessions for phase 2