Web Security Model

Web Security

- making websites safe to visit
- protecting user data across websites
- allowing bad sites to run without interacting with good sites
- supporting secure web applications
- securing internet traffic (e.g., TLS)

URI URN URL

- uniform resource indicator (URI)
  - a unique string meant to identify some resource
- uniform resource name (URN)
  - a kind of URI that gives a unique name
  - urn:isbn:0130460192
  - urn:ietf:rfc:2648
  - urn:lex:eu:council:directive:2010-03-09;2010-19-UE

URI URN URL

- uniform resource locator (URL)
  - most common kind of URI
  - identifies the resource
  - also gives a sequence of instructions on how to get it
- format: schema:[//[user@]host[:port]]path[?query][#fragment]
  - host include `www.ucalgary.ca`
  - schema include http, ftp, https, file
  - ports omitted if default for the schema

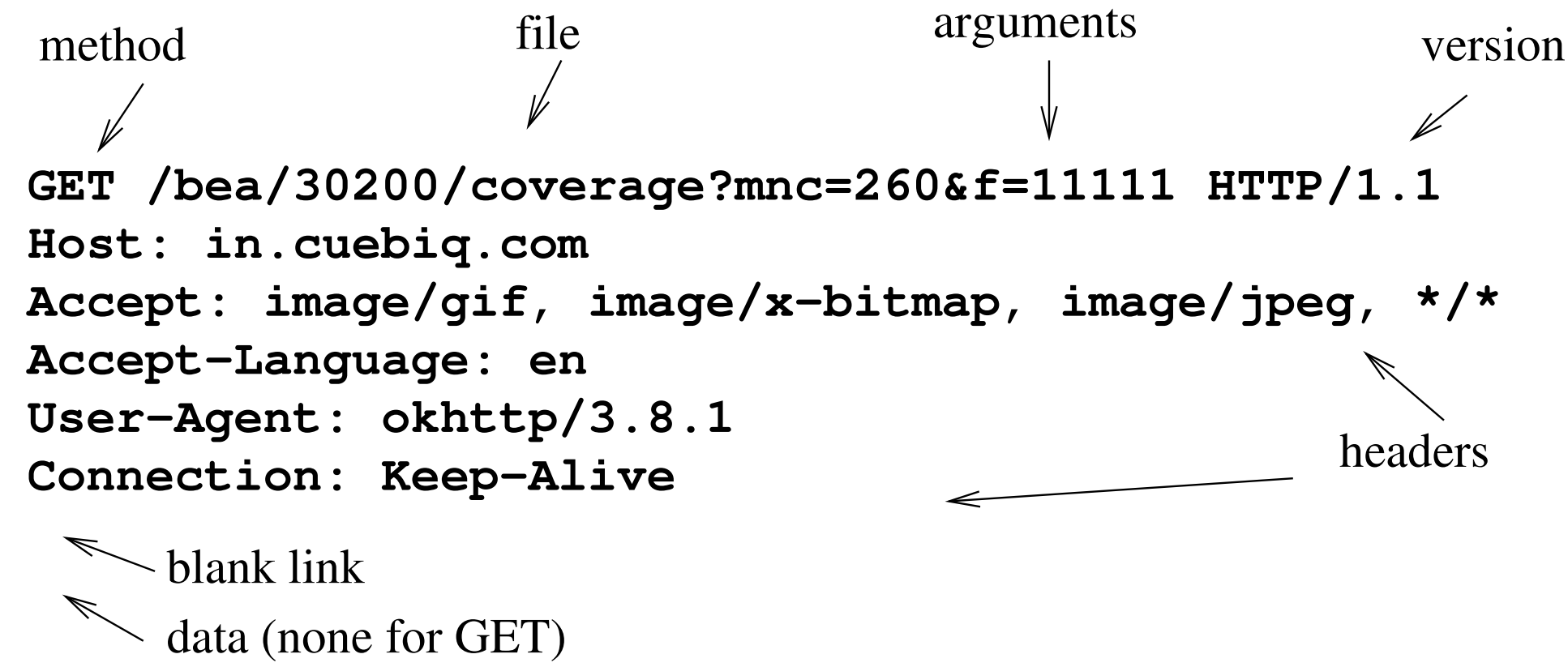
Host or Domain

- a series of one or more dot-separated parts
  - `www.ucalgary.ca`
  - `localhost`
- top level domain (TLD): com, ca, org, net
- second and third level domains are subordinate to their parent
  - .ca is top of the tree, then calgary, then the www server
  - you can add more `myserver.cs.ucalgary.ca`

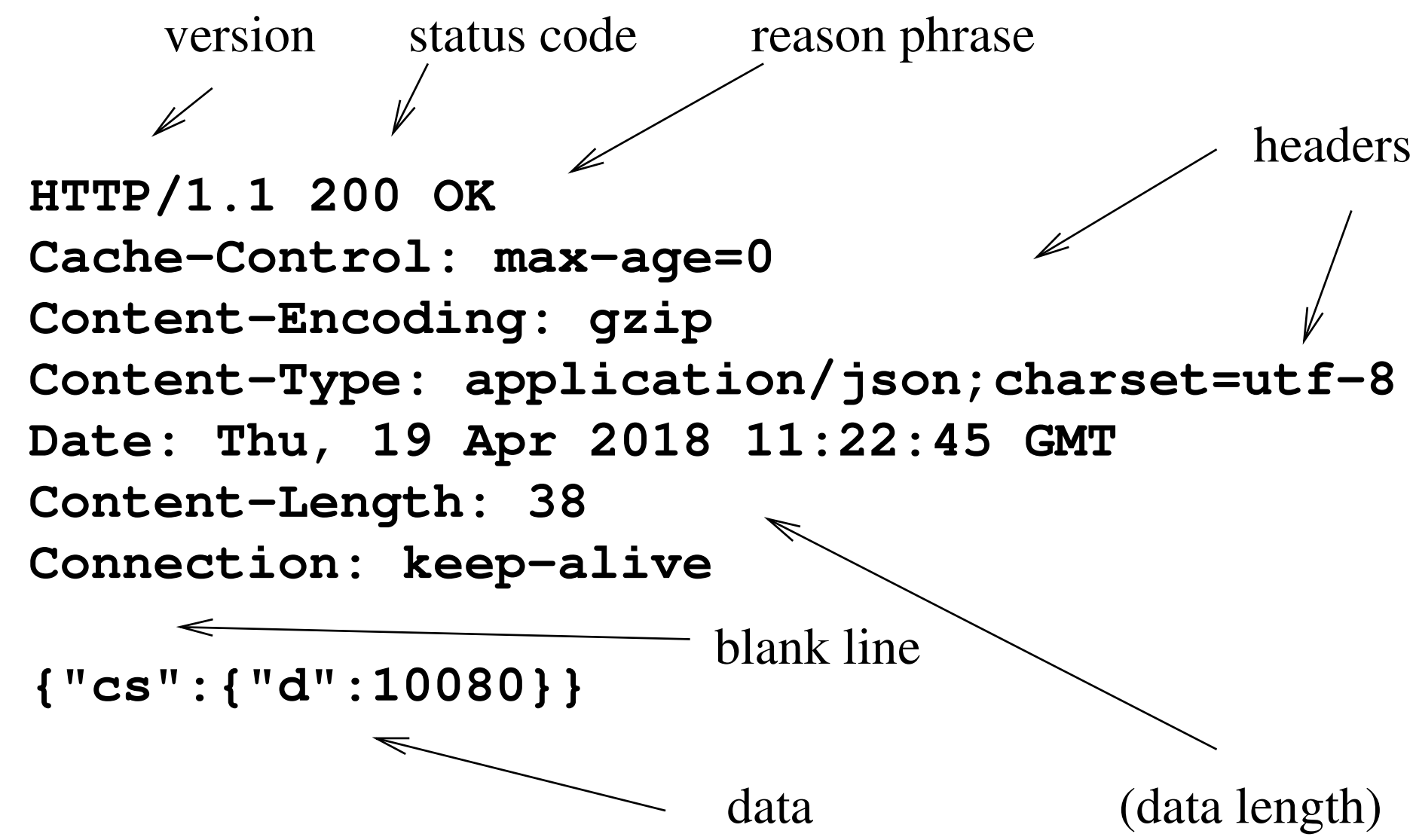
HTTP: HyperText Transfer Protocol

- used to request and return data
  - GET, POST, HEAD, PUT, etc.
- stateless request/response protocol
  - each request is independent of previous requests
  - statelessness has a significant impact on design and implementation of applications

HTTP Request



HTTP Response



Goals of Web Security

- safely browse the Web
  - malicious website cannot:
    - steal information from the user
    - modify legitimate sites
    - otherwise harm the user
- support secure web applications
  - applications delivered over the web should have the same security as local software
  - what would these be?

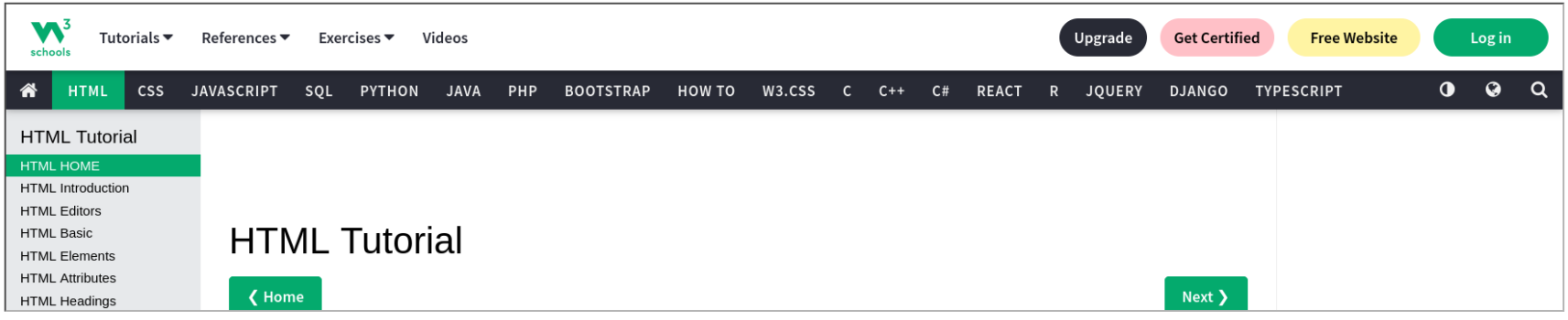
Goals of Web Security

- security should be provided even if the user visits both a good and bad site
- evil sites should not be able to interfere with good sites
- even when they are run
  - at the same time
  - separate window
  - separate tab
  - in an iframe on the same webpage
    - i.e., evil site presents good site inside

HTML Iframes

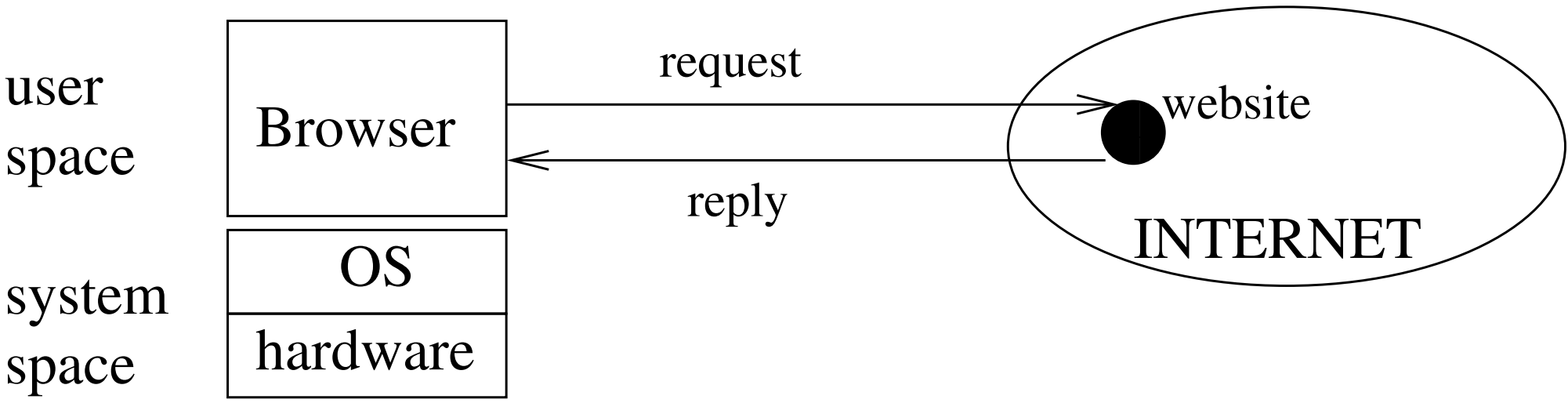
❮ Previous Next ❯

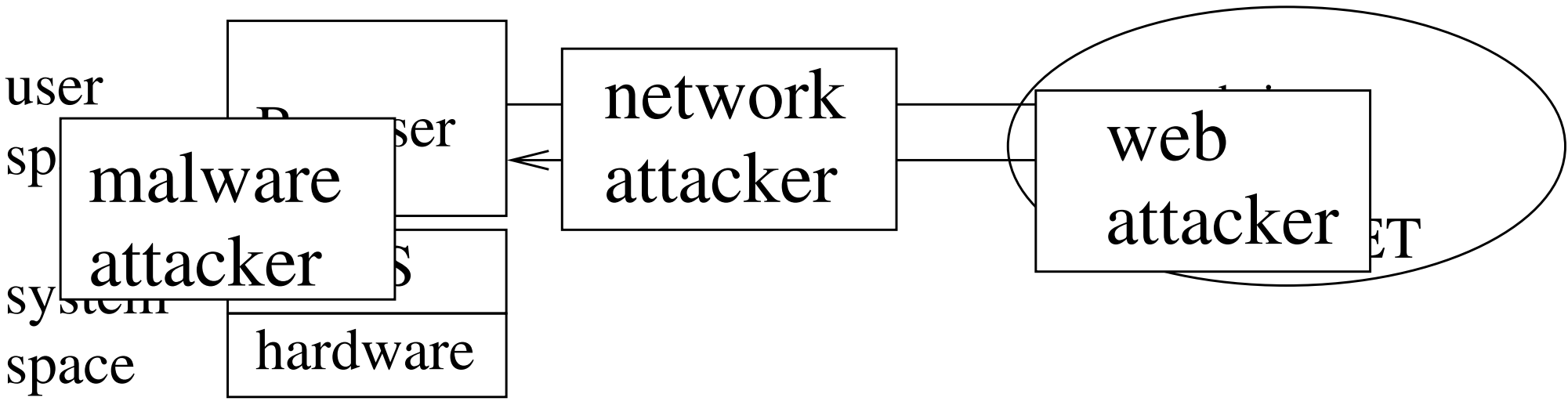
An HTML iframe is used to display a web page within a web page.



Two Sides of Web Security

- web browser
  - responsible for securely confining Web content
  - e.g., evil site iframing a good site must not learn your password as you type it in
- web applications
  - e.g., online merchants, banks, blogs, collaborative editing
  - mix of server-side and client-side code
    - node, PHP, Ruby, python on server side
    - mostly JavaScript on client side
  - many potential bugs: XSS, XSRF, SQL-injection
    - e.g., evil site must not be able to send queries on your behalf even if you logged in through an iframe
    - e.g., evil site should not give a script that looks like it came from someone else
    - e.g., evil user should not be able to change the price of an item when purchasing


user
space
Browser
request
website
reply
INTERNET
system
space
OS
hardware


user
sp
system
space
malware
attacker
network
attacker
web
attacker
hardware

Malware Attacker

- malicious code runs on victim's computer
- can exploit bugs in software
- can convince user to install malicious content
  - e.g., masquerade as anti-virus, video codec, etc.

Network Attacker

- passive attacks
  - wireless eavesdropper
- active attacks
  - evil Wi-Fi router
  - DNS poisoning

Web Attacker

- controls a malicious website (attacker.com)
  - can even have a nice SSL/TLS certificate for the site
- user visits attacker.com
  - phishing email
  - enticing content
    - 10 things that you should see, especially number 7!
  - placed by an ad network
  - clicked by accident
  - tricked into clicking (clickjacking)
- attacker has no other access to user machine

iframe attacker:
iframe with malicious content included
in otherwise honest webpage

iframe attacker:
iframe with malicious content included
in otherwise honest webpage
come from third parties but appear
on the same page as first party

iframe attacker:
iframe with malicious content included
in otherwise honest webpage
come from third parties but appear
on the same page as first party
allows embedding a different
website inside the current one

Browser: Basic Execution Model

- each browser window or frame
  - loads content
  - renders it to the screen
    - processes HTML, CSS, run javascript
    - load images, subframes, etc.
  - respond to **events**
- events
  - user actions like onclick, onkeydown
  - rendering behaviour like onload, error
  - timer elapsing like setTimeout
  - AJAX: dynamic loading of content
    - main model now so name is not mentioned

Document Object Model (DOM)

- HTML is internally represented as a `document` object
  - it has `properties` that are themselves objects
  - these are arranged in a hierarchical structure
  - everything in HTML is put somewhere inside
    - `anchor` objects have `href` properties
- the browser displays the HTML on a **frame**
  - can be a window or part of a window
  - represented by a `window` object
  - the `window.location` property has the URL components as properties
- the `document` object is the root and is standardized by the DOM
  - the DOM is an API for JavaScript to manipulate anything inside
  - e.g., `window.location = ‘‘evil.com’’`

JavaScript

- language executed by the browser
  - scripts are embedded in webpages
  - can be set to run at different times:
    - before loading HTML
    - before viewing page
    - while viewing page
    - when leaving page
- JavaScript allows malicious webpages to **execute code on user's machine**
  - you download code and run it
  - heavily sandboxed to prevent arbitrary behaviour

JavaScript History

- created by Brendan Eich at Netscape
  - scripting language for Navigator 2
- later standardized for cross-browser compatibility
  - ECMAScript
- has nothing to do with java
  - name was part of marketing deal
  - “Java is to JavaScript as car is to carpet”

```
Welcome to Node.js v18.19.1.
Type ".help" for more information.
> [] == ![]
true
> null == 0
false
> null + 1 == 1
true
> false == '0'
true
> 3 + "3"
'33'
> "3" - 5
-2
> "4" * "3"
12
>
```

Uses of JavaScript

- active web
  - change images, hide elements, change cursor
  - dynamic content manipulation, updates
- form validation
  - "credit card field must not have spaces"
    - least surprise failure
    - design failure
    - possible INPUT VALIDATION FAILURE
- lots of web apps
  - collaborative editing, social media, etc.

How can I use credit card numbers containing spaces?

Asked 13 years, 5 months ago Modified 6 years, 10 months ago Viewed 20k times

12

Some fancy websites show an error dialog when it is detected that an untrained shopper has entered a credit/debit card number as it is printed on their card with spaces. Is it possible in some way to write a Java web app that handles these numbers with spaces as if they were correct?

java string usability

51 I simply can't stop myself from pointing out that it's: "An engineer working in Java SE security for Sun." who is asking. Please tell me you're joking, or it's some kind of a hoax... or that it's 4am and you're simply writing-questions in your sleep... – viraptor May 18, 2009 at 0:56

16 Nope. The spaces cannot easily be removed from the input. You can't rearrange user inputs -- that's just doing too much for the user. If a person can't use the software, it's the user's problem. They need to learn to use software by manually removing spaces from a long, incomprehensible string of random digits. – S.Lott May 18, 2009 at 1:12

80

My view is that any Web app that rejects a credit card number with spaces isn't doing its job. When you receive a credit card number, it's easy enough to do:

```
String ccNumber = ccNumber.replaceAll("[\\s-]+", "");
```

to remove spaces and dashes (some use those too). Then validate the result. You'll simply annoy your users if you force them to remove spaces you could just as easily do.

Where to put JavaScript

- embedded in HTML as a <SCRIPT> element
  - <script> alert("Hello, world!") </script>
  - <script type="text/JavaScript" src="notmalicious.js" />
- event handler in a tag
  - <a href="http://www.ucalgary.ca" onmouseover="alert('click');">
- JavaScript schema
  - click here

JSON

- JavaScript Object Notation
  - serialization format for pure data
  - null, ints, floats, strings, arrays, and objects
  - dominant exchange format for HTTP
- object: {key: “value”, number: 2}
- array: [3, “hello”, 12.34, NaN, null, {point: [3, 6] }]

```
{
  "input": {
    "actor_id": "0",
    "app_use_state": "FOREGROUND",
    "bluetooth_info": {
      "enabled": "true",
      "scan_results": [
        {
          "advertisement_payload_base64": "0201061bff0118beac6269746579626974657962697465796269746579f401",
          "age_ms": "5548",
          "hardware_address": "AB:B1:E7:7E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201061aff4c00021501022022fa0f010000acdd1c6502da1cd0e7a64bf4",
          "age_ms": "5495",
          "hardware_address": "AB:B1:E6:6E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201060303aafe1116aafe10f4006c6f6f73656c6174636807",
          "age_ms": "5445",
          "hardware_address": "AB:B1:E8:8E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201061bff0118beac6269746579626974657962697465796269746579f401",
          "age_ms": "5343",
          "hardware_address": "AB:B1:E3:3E:1B:BA",
          "rssi_dbm": "-12",
        },
      ],
    },
    "cell_info": {
      "connected": [
        {
          "is_network_roaming": "false",
          "network countru iso": "ca",
```

Same Origin Policy (SOP)

Same Origin Policy (SOP)

SOP is an isolation and access control philosophy to protect data connected to one host being accessed by another (possibly malicious) host

Same Origin Policy

- avoid having an evil website:
  - read data from another website
  - manipulate data from another website
- SOP is a compromise
  - one option would be to not allow any content other than the website itself
    - e.g., embed all files and scripts into the single HTML page
  - one option would be to not allow any third-party content
  - SOP allows third-party content, but sandboxes it
    - ideally treats it as though it were another opened tab
    - considered a vulnerability when not true
- SOP controls all access to the DOM

Same Origin Policy

- a base HTML document is assigned an **origin**
  - based on the URI that retrieved it
  - consists of (scheme, host, port) triple
    - **scheme**://**host**:**port**/irrelevant/to/sop
  - two origins are the same if the whole triple matches
    - string matching

Same Origin?

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

http://wikipedia.org/ and http://www.wikipedia.org/

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

http://wikipedia.org/ and http://www.wikipedia.org/

http://wikipedia.org/ and https://wikipedia.org/b/

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

http://wikipedia.org/ and http://www.wikipedia.org/

http://wikipedia.org/ and https://wikipedia.org/b/

http://wikipedia.org:81/ and http://wikipedia.org:82/

SOP assigns every component in DOM an origin based on what loaded it into the webpage

SOP assigns every component in DOM an origin
based on what loaded it into the webpage
images, scripts, content get the **origin** of loader

SOP assigns every component in DOM an origin
based on what loaded it into the webpage
images, scripts, content get the **origin** of loader
the content of an iframe has the
origin of the URL that serves the iframe;
not the website that embeds it.

SOP: one origin should not be able to access the resources of another origin.

SOP: one origin should not be able to access the resources of another origin. JavaScript on one iframe cannot read or modify pages from different origins

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP or status of “being logged in”

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP or status of "being logged in"
e.g., library, email account, LAN resource (router config)

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP or status of "being logged in"
e.g., library, email account, LAN resource (router config)
should this other website be able to read that?

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP or status of "being logged in"
e.g., library, email account, LAN resource (router config)
should this other website be able to read that?
or modify how it looks?

Web Cookies

HTTP is Stateless

HTTP is Stateless
each HTTP request is a standalone event

HTTP is Stateless

each HTTP request is a standalone event

this is a poor match to how it is used

HTTP is Stateless

each HTTP request is a standalone event

this is a poor match to how it is used

e.g., shopping carts, being “logged-in”, language preferences

Web cookies add state

Web cookies add state

a cookie is a file created by a website to store information in the browser

Web Cookies

- server replies with an HTTP header Set-Cookie
  - e.g., JSESSIONID=4F5627840794D2E93F9D083EDFD9F263; Path=/; HttpOnly
- cookie has parameters
  - "argument=value" format or just "argument"
  - arguments are optional
  - arguments are separated by semicolon-space

SOP for Cookies

- cookies have a different SOP than the DOM's SOP
  - can be sent on any port
  - can be sent using HTTP or HTTPS
  - must be sent to same domain **and path**
    - subdirectories okay
- these policies can be tweaked

Web Cookie Arguments

- Secure; (only send over encrypted channel)
- HttpOnly; (deny DOM API access to cookie)
- Expires=Mon, 29-Jan-2018 14:30:11 GMT; (date to delete)
  - format is Wdy, DD-Mon-YYYY HH:MM:SS GMT
  - past date means delete immediately
  - no expire means this session only

Web Cookie Arguments

- Domain=VALUE
  - by default cookie can only be read by exact domain
    - e.g., myserver.cs.ucalgary.ca
  - is used to **widen** domain
    - e.g., .ucalgary.ca
    - means anything that ends in .ucalgary.ca
    - is this secure? what is threat model?
  - cannot be used to allow different registerable domain
    - "same-site"
- Path=VALUE;
  - widen path of who may read the cookie
  - by default only HTML in the current directory and subdirectories can get cookie

CNAME

- third parties can appear to be same-site if the first party allows
- e.g., smetrics.cbc.ca CNAMEs to cbc.ca.102.122.2o7.net
  - this is an adobe IP for their analytics system
- allows cbc.ca cookies to be sent to smetrics.cbc.ca if domain is widened

Cookie Uses

- session ID
  - random number stored in a cookie and sent to the server
  - indexes server-side state for the client
- authentication
  - cookie's session ID also acts as an authenticator
    - the cookie proves to the website that the client previously authenticated correctly
  - what does this remind you of?
  - **cookie theft**: anyone knowing this number can impersonate you
    - why can't it prove that it is Alice?

SOP motivation

SOP motivation

if a script loads a website in an iframe

it may be appear different based on cookie

SOP motivation

if a script loads a website in an iframe

it may be appear different based on cookie

e.g., it is logged into to website

SOP motivation

if a script loads a website in an iframe

it may be appear different based on cookie

e.g., it is logged into to website

SOP prevents script from reading that iframe

Cookie Uses

- personalization
  - helps the website recognize the user from a previous visit
  - store settings locally or remotely
- tracking
  - follow the user from site to site
  - learn their browsing behaviour, etc.
- third-party cookies
  - a.com has an iframe for b.com
  - b.com sends a cookie for the site
  - user will send that back to b.com whenever visiting a.com

Cookies and Site Data

Your stored cookies, site data and cache are currently using 760 KB of disk space. Learn more

Clear Data...

Manage Data...

Exceptions...

- (●) Accept cookies and site data from websites (recommended)
  - Keep until: Firefox is closed
  - Accept third-party cookies and site data: Never
- ( ) Block cookies and site data (may cause websites to break)