Increasing subjective testability with Fiddler

Rasmus Koorits
RSI

@rkoorits

Get Fiddler 4

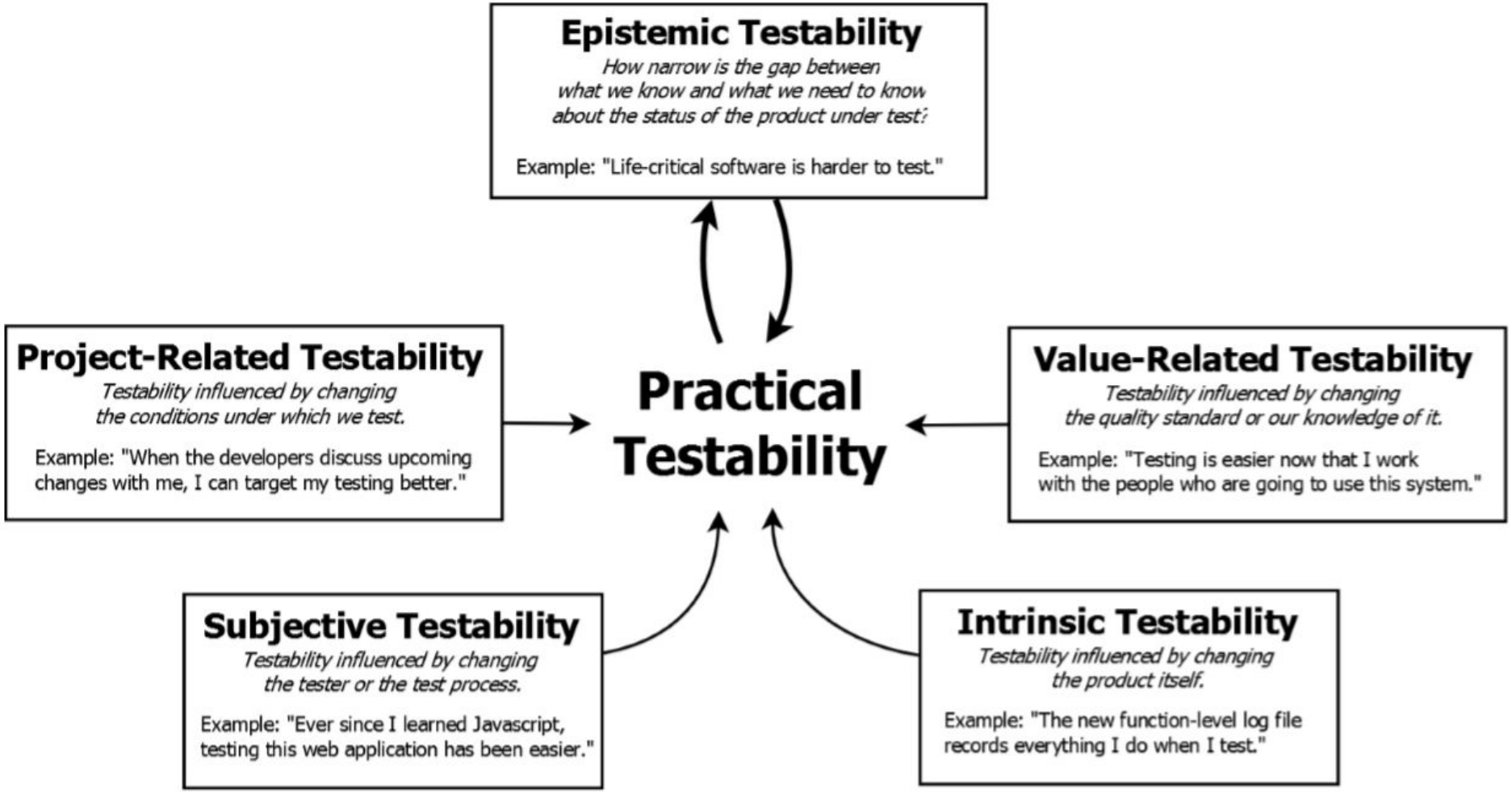
- Get Fiddler from https://www.telerik.com/download

Structure of the workshop

- Explain what I mean by “Increasing subjective testability”
- Cover some basic concepts
- Explain how fiddler works
- Switching between theory segments and practice sessions
- Ask questions at any point!
- Coffee break somewhere in the middle
- Share your testability issues and we can solve them together

On Testability

- Two cool models!



http://satisfice.com/tools/testability.pdf

On Testability

- Visibility
  - How can we see what is going on inside the system
  - How can we know if what we did had the expected result
  - … and did not have unexpected results
- Controllability
  - How can we alter the state of the system
  - Or the data in the system

*RST materials and http://satisfice.com/tools/testability.pdf

On Testability

- Proxy servers can help with both Visibility and Controllability when testing in a client/server environment!
- … but we need some practice in order to realize when to use them.

Pattern recognition for tool selection

- The ability to recognize a familiar problem
- And picking the appropriate tool (or technique)

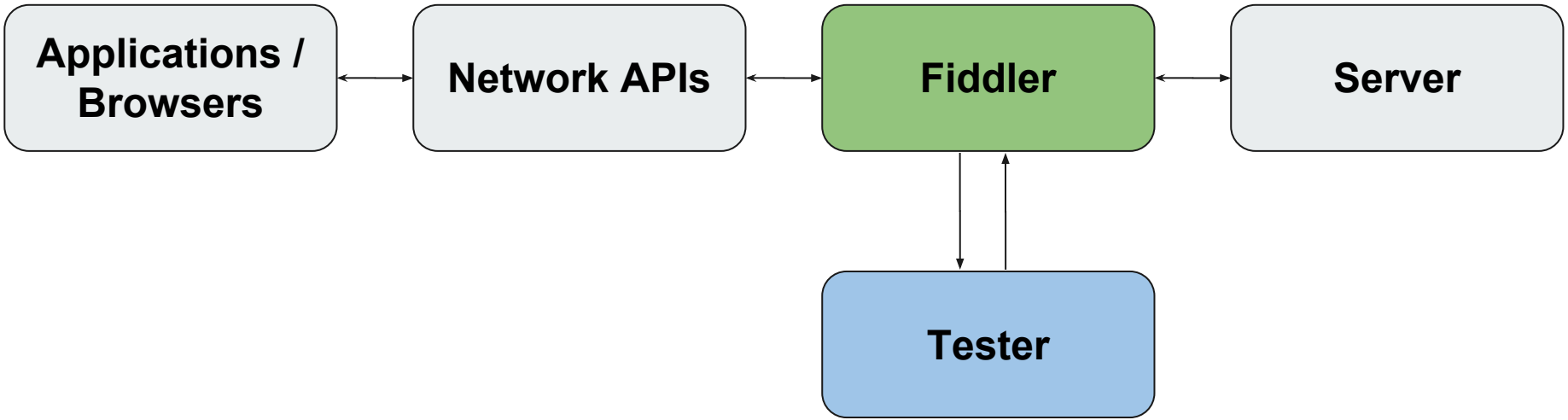
Pattern recognition for tool selection

- A loose screw
- Leaking pipe
- Testing the maximum size of a large text field
- Testing a text editor

Goals of this workshop

- Learn about problem patterns that proxy tools can help with
- Gain hands-on experience to facilitate the above
- Try to solve some real life testability problems as a group
- You start using proxy servers in your daily work

What is a Proxy?



Any questions so far?

HTTP(s) Primer - Request/Response paradigm

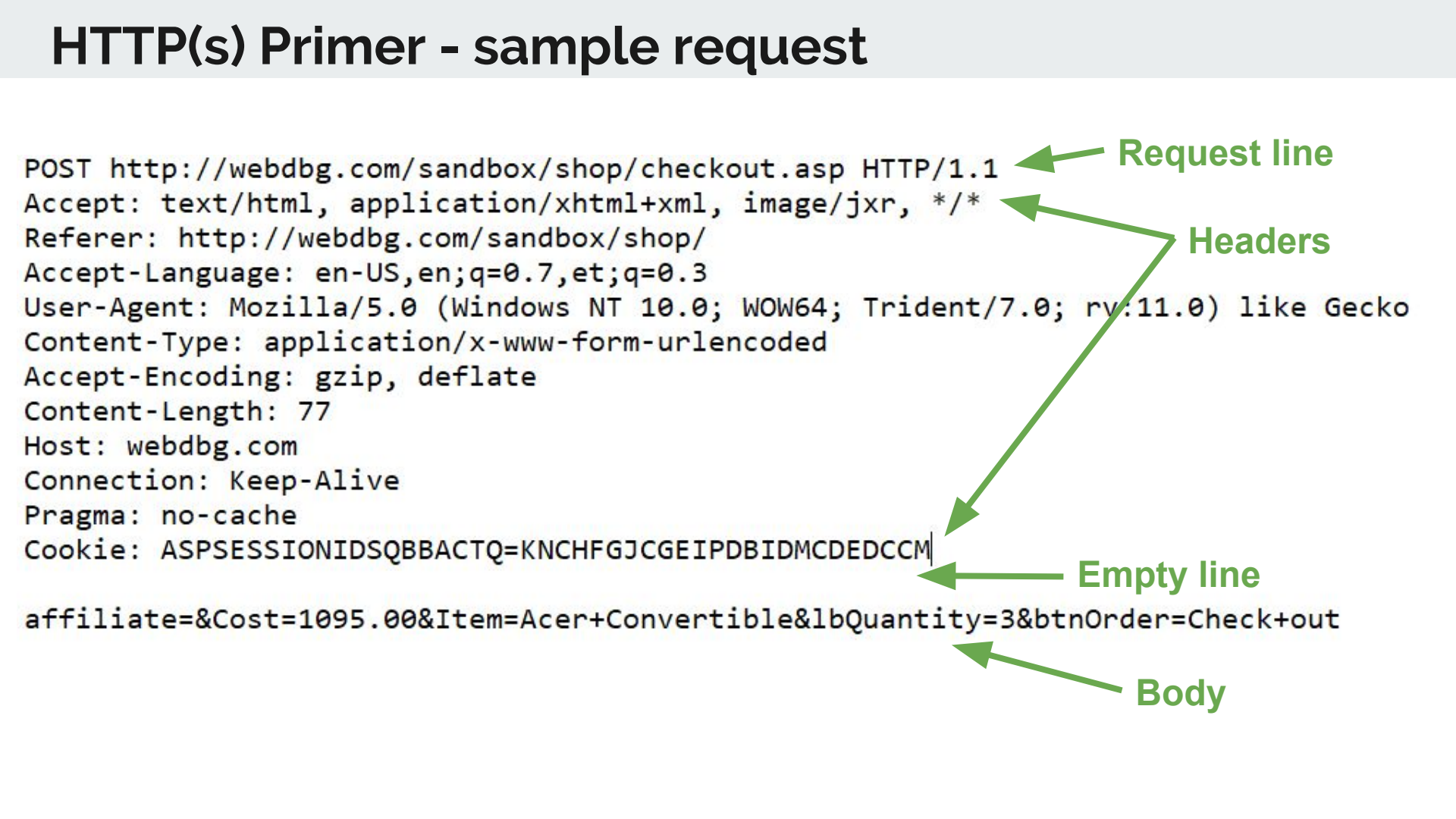
- Clients make a **Request**
- Servers return a **Response**
- … thus forming a request/response pair
- Fiddler calls this a Session
- (Except for websocket connections, which we will cover later)

Web apps go to a great lengths to hide the basic paradigm!
(because it just looks better that way - consider chatrooms or email clients)

HTTP(s) Primer - sample request

```
POST http://webdbg.com/sandbox/shop/checkout.asp HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://webdbg.com/sandbox/shop/
Accept-Language: en-US,en;q=0.7,et;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Content-Length: 77
Host: webdbg.com
Connection: Keep-Alive
Pragma: no-cache
Cookie: ASPSESSIONIDSQBBACTQ=KNCHFGJCGEIPDBIDMCDEDCCM

affiliate=&Cost=1095.00&Item=Acer+Convertible&lbQuantity=3&btnOrder=Check+out
```

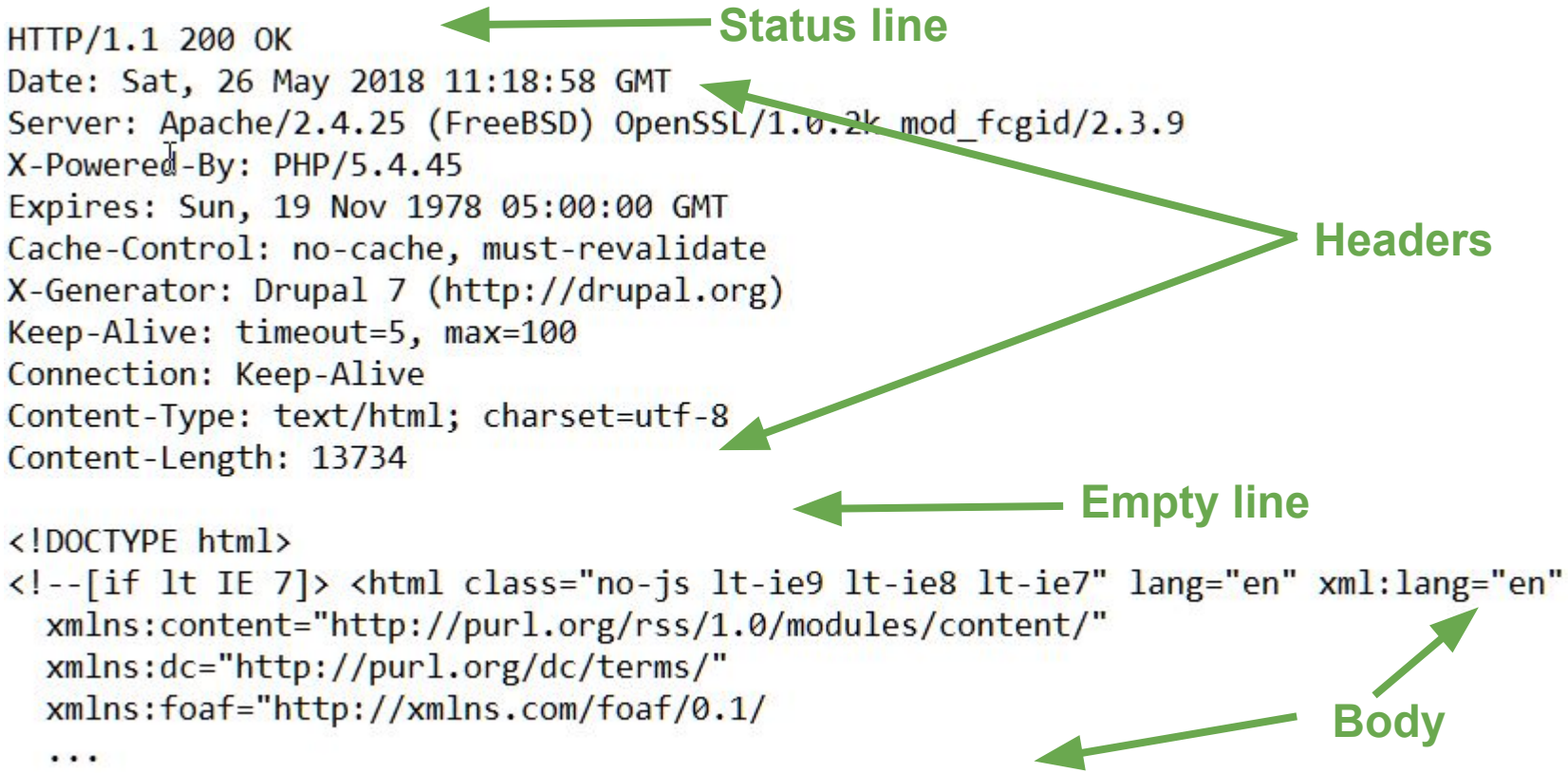


HTTP(s) Primer - Request Methods

- GET
- POST
- PUT
- DELETE
- OPTIONS
- …

But these are at *least partially* arbitrary

HTTP(s) Primer - sample response



HTTP(s) Primer - Response codes

- 200 ⟵ OK
- 404 ⟵ Not Found
- 418 ⟵ I’m a Teapot!
- 500 ⟵ Internal Server Error

But these are *sometimes* arbitrary as well!

… But some tools/frameworks/environments make strong assumptions

HTTP(s) Primer - Response Body

- HTML
- JSON
- XML
- .. or anything else

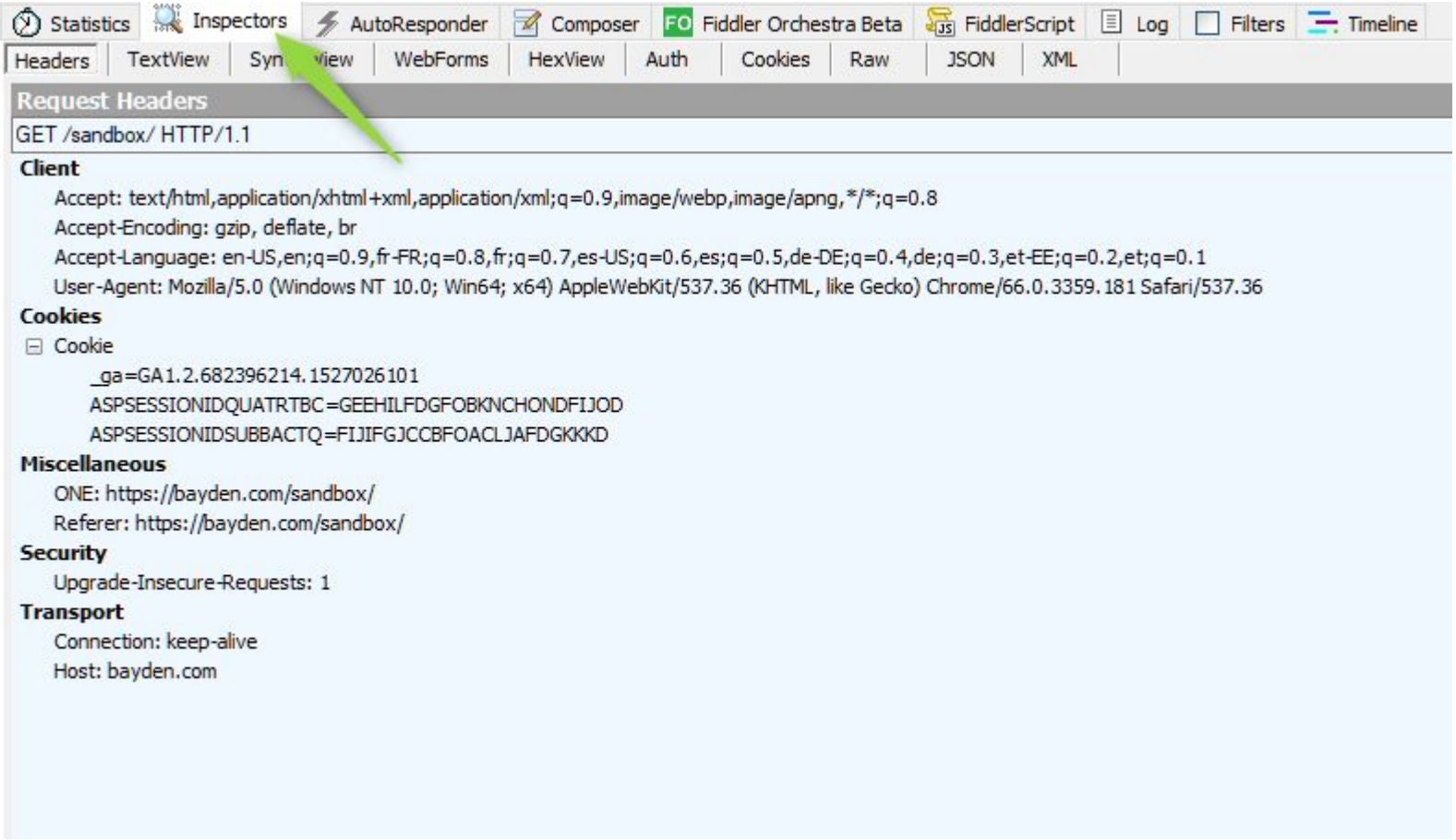
Any questions so far?

Team up!

Let’s run Fiddler!

- As well as Chrome/IE/Edge
- Brace for various warnings at first

Inspectors tab



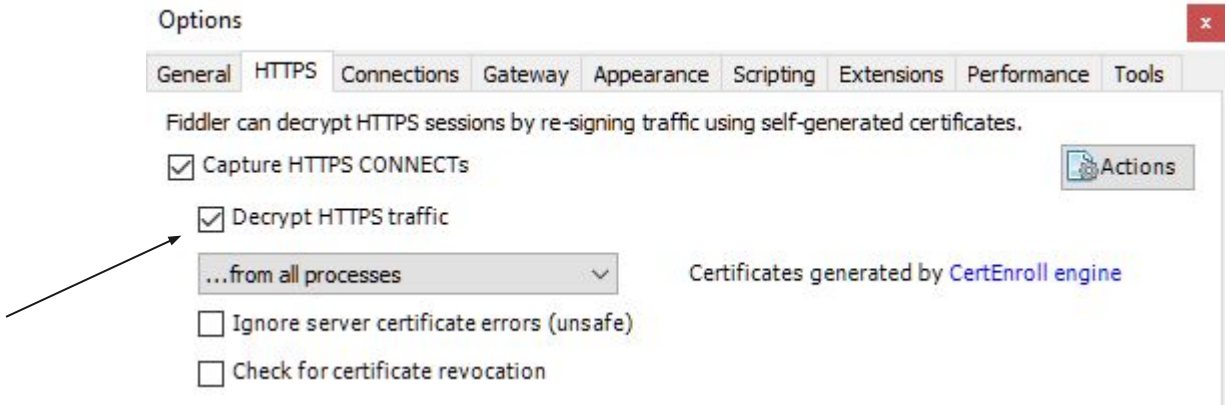
Let's Fiddle stuff!

- **Exercise:** view requests/responses at the Fiddler sandbox - https://bayden.com/sandbox/
  or at the sandbox created for this workshop:
  http://ntd.codeandtest.org

- Try out the various Inspectors on different sessions

Enabling HTTPS decryption

File menu: Tools -> Options



See https://en.wikipedia.org/wiki/Man-in-the-middle_attack

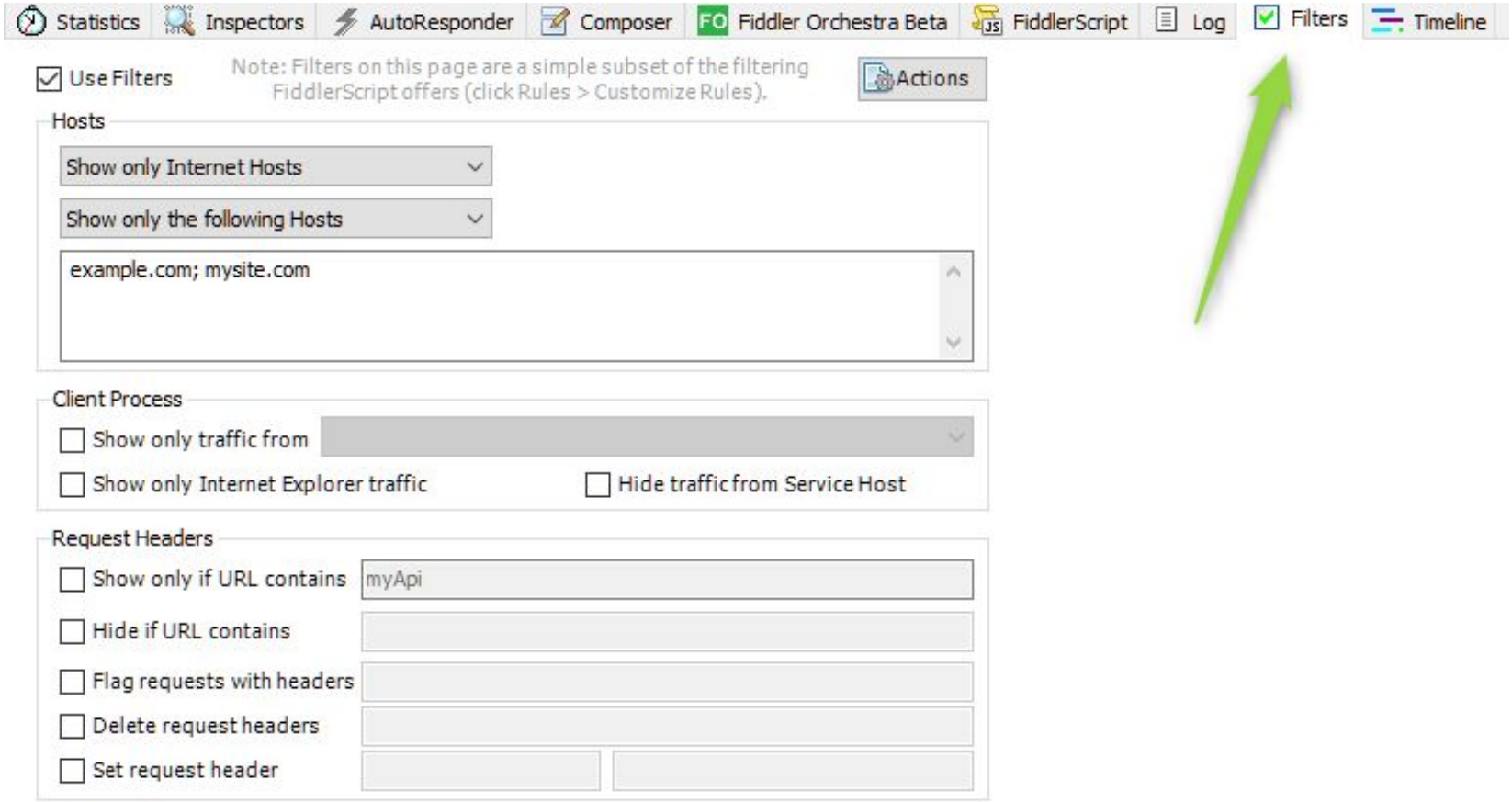
Why not just use Chrome Dev Tools?

- Strange Chrome quirks
- Cross browser testing
- Desktop applications
- Phone apps!
- The power of Fiddler!

Filtering out the noise

- Filters tab
- Quick exec
  - bold
- FiddlerScript

Filtering out the noise



Let's Fiddle stuff!

Exercise: Filter to only show requests/responses that have the relevant host names

You can use https://ntd.codeandtest.org as a sandbox

Sometimes this is a bad idea, as unexpected calls will go unnoticed!

Breakpoints!

- Tampering Requests before they reach the Server
- Tampering Responses before they are returned to the Client
- Can be enabled from various places:
  - QuickExec: bpu, bpa
  - Filters
  - FiddlerScript
  - ....

Let's Fiddle stuff!

Exercise 1: Change something in a request

- For example, turn a POST request into a GET request

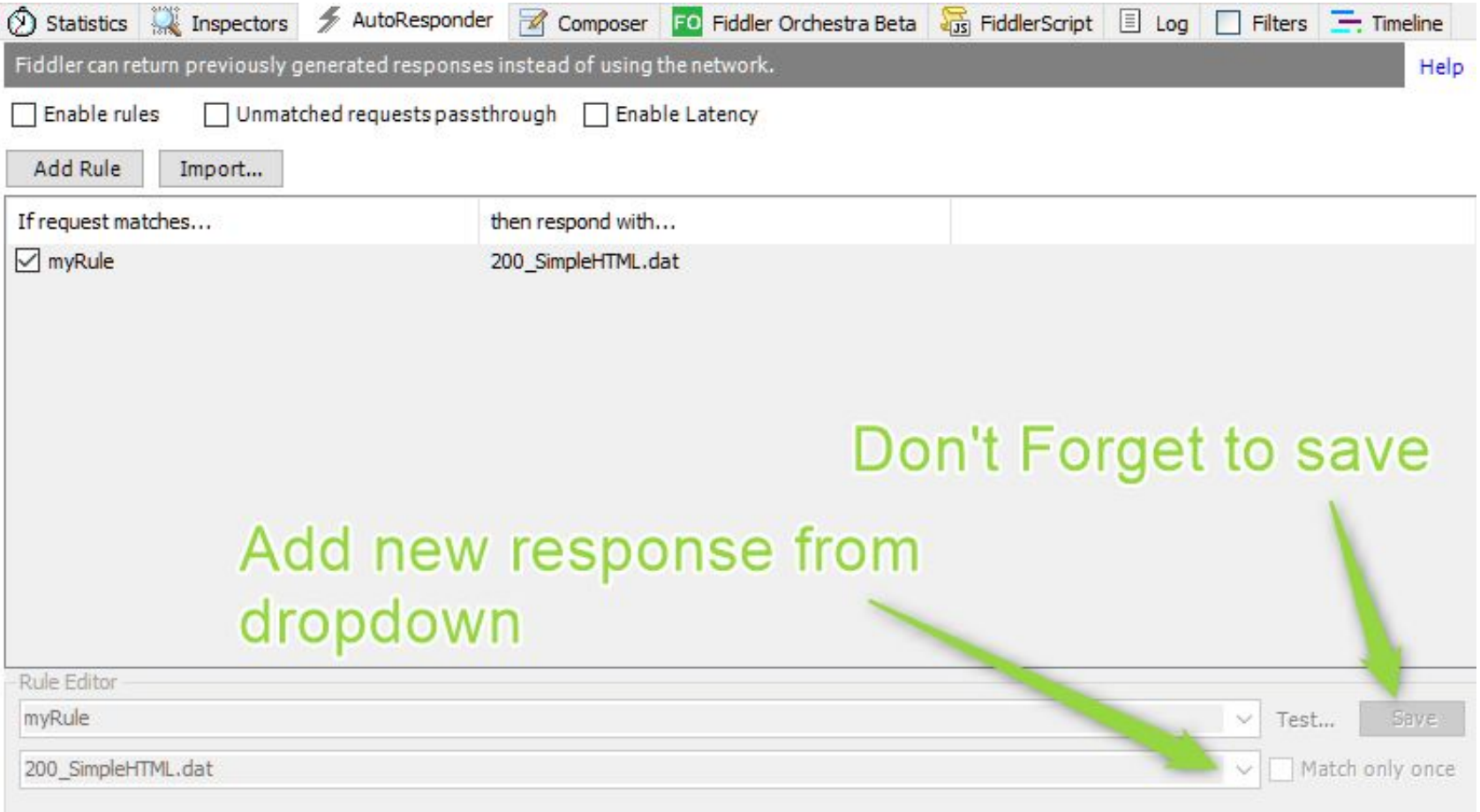
Exercise 2: Change something in a response

- Turn a 200 OK response into a 500 Error

AutoResponder

- Overwrite a server response
  - With a previously saved response
  - Or a handcrafted one
  - Or even a picture!

AutoResponder tab



Let's Fiddle stuff!

Exercise 1: Replace a 200 response with a 500 error

Pattern: We want to test how the front end handles an error

Hint: use any page or ntd.codeandtest.org for replacing
Hint: example error responses can be seen at http://getstatuscode.com/500

Let's Fiddle stuff!

Exercise 2: Replace an image with another image

Pattern: We want to switch out any file (such as a new version of a css file or javascript library) to see how it works in production

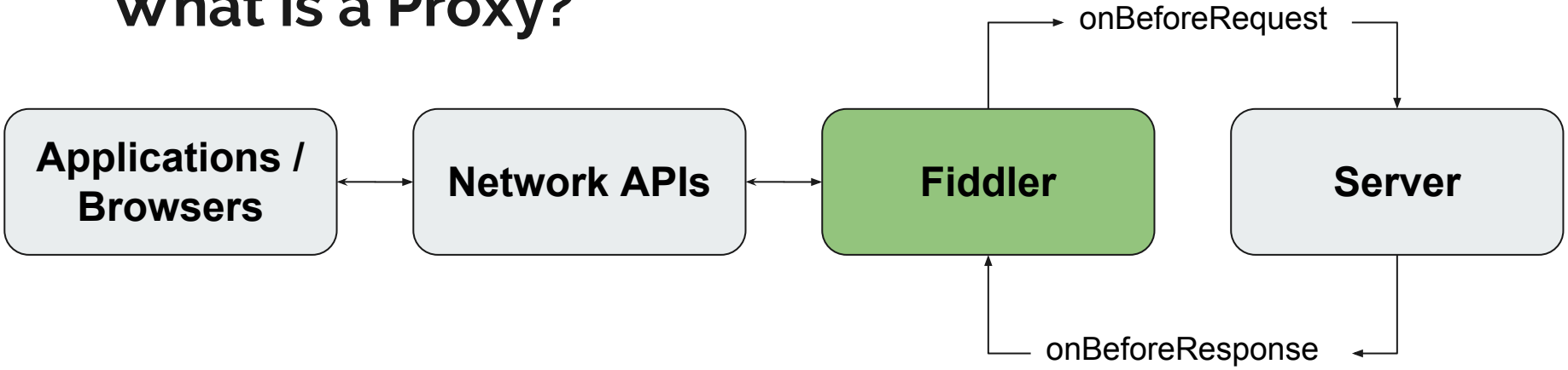
Hint: use any page or the image page at ntd.codeandtest.org

Any questions so far?

Custom Rules / FiddlerScript

- It’s pretty much javascript!
  - Except it's completely different
- https://docs.telerik.com/fiddler/KnowledgeBase/FiddlerScript/
- We mostly operate inside two existing methods:
  - OnBeforeRequest
  - OnBeforeResponse
- Note: ctrl+s does not work if opened from right hand menu

What is a Proxy?



FiddlerScript

```
if (oSession.host == "codeandtest.org"
    && oSession.url.Contains("hello"))
{
    oSession["ui-color"] = "purple";
    oSession["ui-bold"]= "test";
}
```

Let's Fiddle stuff!

Exercise 1: Change the session color for a particular url or host/path

Exercise 2: Rewrite the request url to point at another url

Exercise 3: Rewrite the response code

Exercise 4: Rewrite the response body

Exercise 5: Add a delay to the request or response

Let's Fiddle stuff!

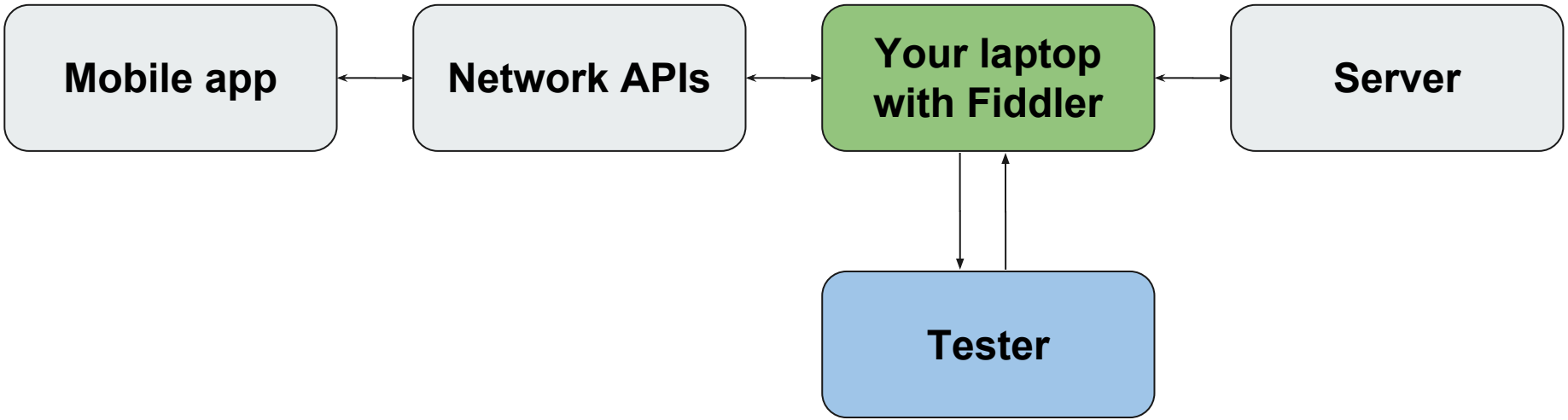
Exercise 6: Variable delay - make all requests with a specific path take between 0 and 10 seconds longer than they should.

Exercise 7: Intermittent failure recovery - make 10% of requests return an error and an empty body.

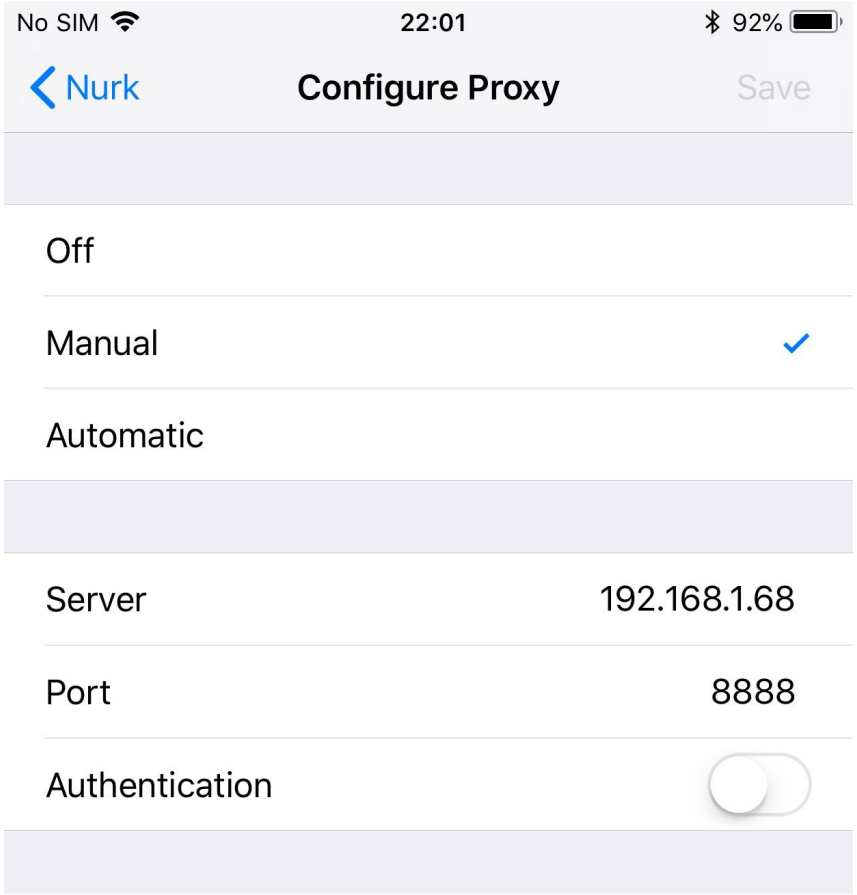
Proxy for other devices

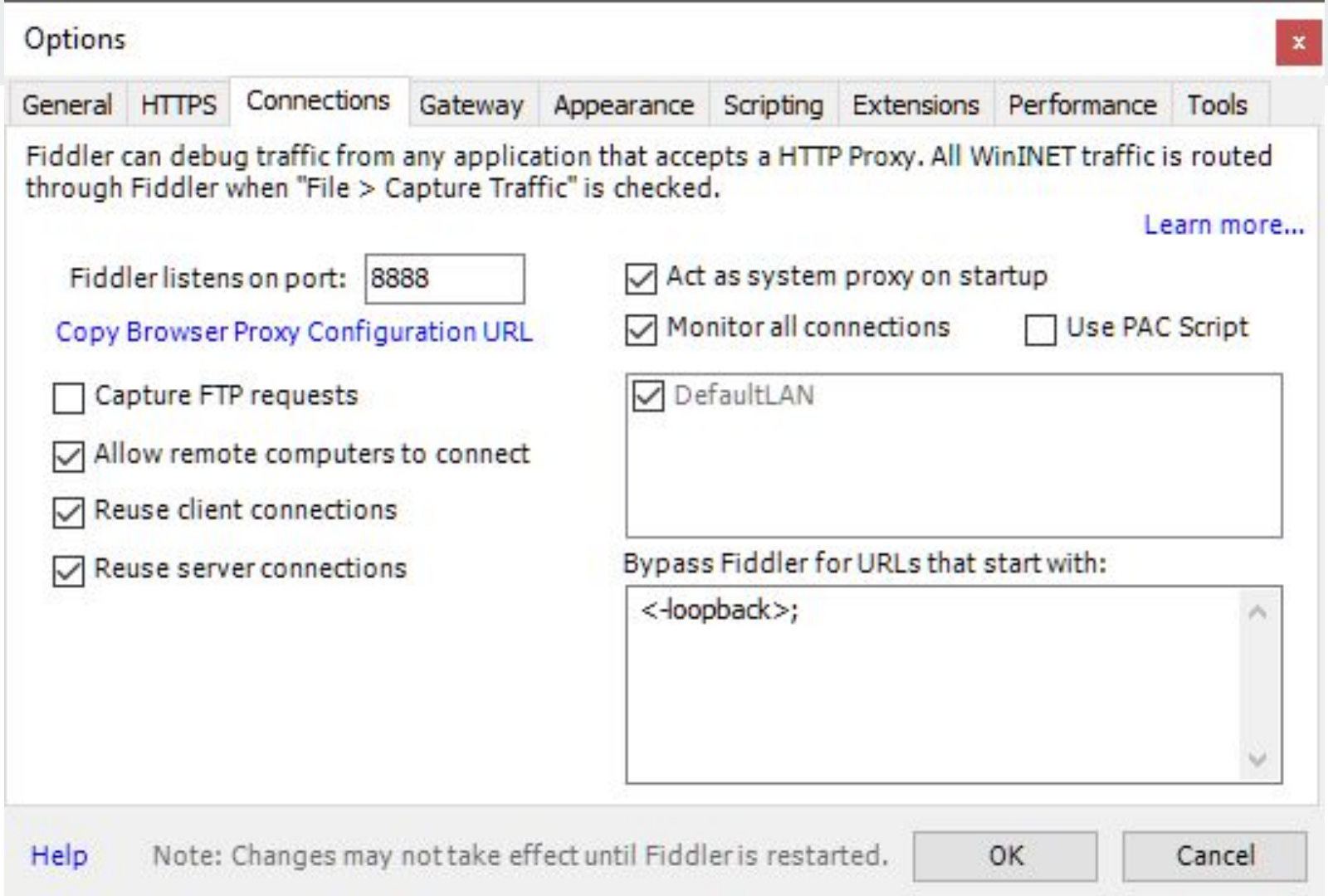
- Setting it up
- Android limitations
- iOS limitations

What is a Proxy?



Proxy for other devices




Options
General
HTTPS
Connections
Gateway
Appearance
Scripting
Extensions
Performance
Tools
Fiddler can debug traffic from any application that accepts a HTTP Proxy. All WinINET traffic is routed through Fiddler when "File > Capture Traffic" is checked.
Learn more...
Fiddler listens on port: 8888
Act as system proxy on startup
Copy Browser Proxy Configuration URL
Monitor all connections
Use PAC Script
Capture FTP requests
DefaultLAN
Allow remote computers to connect
Reuse client connections
Reuse server connections
Bypass Fiddler for URLs that start with:
<-loopback>;
Help
Note: Changes may not take effect until Fiddler is restarted.
OK
Cancel

Let’s Fiddle stuff?

Exercise: Point your smartphone at Fiddler running on your laptop

Let’s discuss your use cases!

Questions

Thank you!