Hit Counters

- file:hitcounto.php


```
<title>Visit counter</title>
</head><body><div class="bigger"> <div class="c2">
<?php print updateAndGetCount(100); ?>
</div></div><?php  function updateAndGetCount($iidd) {
    $servername = "localhost";
    $username = "gtstudent";
    $password = "";
    $dbname = "count";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $sql = "SELECT max(count) as c FROM count WHERE id = $iidd";
    $result = $conn->query($sql);
    $row = $result->fetch_assoc();
    if ($row==null || $row["c"]==null) {
       $visits = 1;
    }
    else {
        $visits = $row["c"];
        $visits++;
    }
    $sql = "INSERT INTO count (id, count) VALUES ( $iidd , $visits);";
    $result = $conn->query($sql);
    $conn->close();
    return $visits;
}  ?>
</body> </html>
```

Improvement to Hit counting

- Should not be the whole page
- Should not just insert into table as that will accumulate a lot of junk
  - file: hitcount.php
    - at least addresses junk problem
    - might be better to use SQL update
      - file: hitcount2.php
- Better (quicker) if SQL table had id as primary key
  - id could not be a primary key until after hitcount2.

```
create table countp1 (
        id int NOT NULL,
        count int not null,
        primary key (id)
);

create table countp2 (
        id int NOT NULL primary key,
        count int not null
);

create table countp3 (
        id int NOT NULL,
        count int not null,
        constraint pkc
             primary key (id, count)
);
```

Still better HitCount

file:pagewithhitcount.html

- Rather than making the whole page a hit count, create a small bit to do the hitcount
- Use javascript fetch to get data from PHP
- The html and the hitcount do not have to come from the same place!
  - protocols must agree!
- PHP does not generate HTML, just returns the number of hits
- Page largely separated from counter

```
        <script>
            const baseURL = "http://comet.cs.brynmawr.edu/~gtowell/380/Lec10/";
            function getHit() {
                const data = { id: 12 };
                let fd = new FormData();
                for(var i in data){
                    fd.append(i,data[i]);
                }
                fetch(baseURL+"hitcountwidget.php", {
                        method: 'POST',
                        mode:"cors",
                        body: fd,
                }).then(function(response) {
                    response.text().then(function(text) {
                        console.log($("div.hcc").text() + text);
                        $("div.hcc").text("HC:"+text);
                    });
                });
            }

            $(document).ready(function() {
                getHit();
            }
        </script>
    <div id="12" class="hcc"></div>
    <div style="height:calc(100% - 50px); margin-top:0px”> ….
```

PHP side of a better hitcount

- No awkward HTML
  - better separation of presentation and preparation
- only response is a number
- code here is otherwise the same as hitcount2.php

```php
<?php function updateAndGetCount($iidd) {
    $servername = "localhost";
    $username = "gtstudent";
    $password = "";
    $dbname = "count";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $sql = "SELECT count as c FROM count WHERE id = $iidd";
    $result = $conn->query($sql);
    $row = $result->fetch_assoc();
    if ($row==null || $row["c"]==null) {
        //echo "new counter";
        $visits = 1;
        $sql = "INSERT INTO count (id, count) VALUES ( $iidd , $visits);";
        $result = $conn->query($sql);
    }
    else {
        $visits = $row["c"];
        $visits++;
        $sql = "update count set count=$visits where id=$iidd";
        $conn->query($sql);
    }
    $conn->close();
    return $visits;
}
echo updateAndGetCount($_REQUEST["id"]);

?>
```

Javascript Fetch

- Fetch is a Promise
- First “then” occurs on receiving headers
- In this case body might contain JSON or plain text
  - So examine headers to determine what the body will contain.
    - Invoke a new Promise to get the body of the response and parse appropriately
      - THEN handle the parsed result.

```
fetch(myRequest).then(function(response) {
  const contentType = response.headers.get("content-type");
  if (contentType && contentType.indexOf("application/json")
    return response.json().then(function(json) {
      // process your JSON data further
    });
  } else {
    return response.text().then(function(text) {
      // this is text, do something with it
    });
  }
});
```

More Promising

```
<html>
    <head>
        <script src="../JQ/jquery-1.9.1.min.js"></script>
    </head>
    <body>
        <div id="countout"></div>
        <button onclick="push()" id="mybutton">Push Me</button>
        <script>
            var pushCount=0;
            $(document).ready(function() {
                $("#countout").html("Count " + pushCount);
            });
            function push() {
                gtsleep2(1000).then(function(val) {
                    pushCount++;
                    $("#countout").html(val + " " + pushCount);
                },
                function(reason) {
                    $("#countout").html("Rejected " + reason + " " + pushCount);
                });
            }
            function gtsleep2(ms)
            {
                return(new Promise(function(resolve, reject) {
                    setTimeout(function() { resolve("success"); }, ms);
                }));
            }
        </script></body></html>
```

Two functions in then

depending on call to resolve

or reject in promise

file:eventloop3.html

Best yet HitCount

- Put all of the javascript and supporting CSS into hitcountscript.js and hitcountstyle.css
- Then user only needs to add an element with an attribute hitcountid (along with <link and <script)
  - With a little work could put all css into js file
  - With a little more work, no JQuery

file: pagewithhitcount2.html

```
<html>
    <head>
        <script src="../JQ/jquery-1.9.1.min.js"></script>
        <link rel="stylesheet" href="hitcountstyle.css">

    </head>
    <body>
        <script src="hitcountscript.js"></script>
        <div hitcountid="12" class="hcc"></div>
        rest of page
```

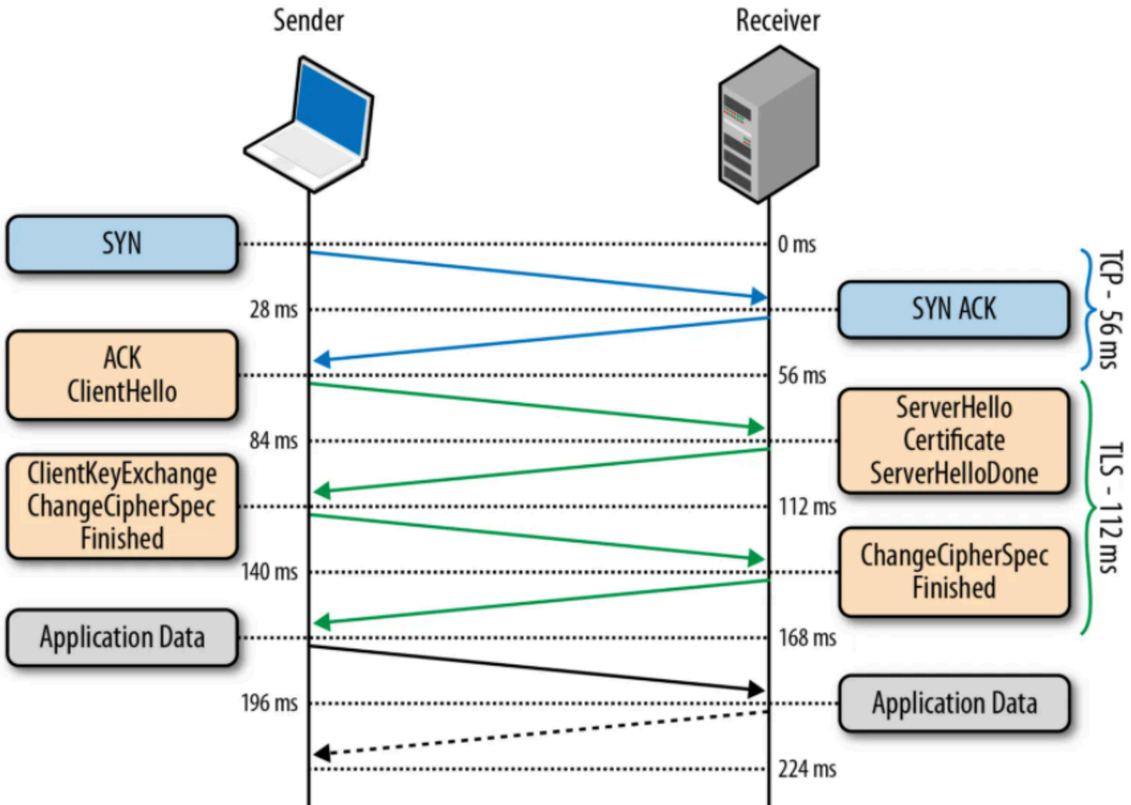
```
function getHit() {
    const data = { id: $("div.hcc").attr("hitcountid"), uurl: window.location.href};
    let fd = new FormData();
    for(var i in data){
        fd.append(i,data[i]);
    }
    fetch(baseURL+"hitcountwidget.php", {
        method: 'POST',
        mode:"cors",
        body: fd,
    }).then(function(response) {
        response.text().then(function(text) {
            console.log($("div.hcc").text() + text);
            $("div.hcc").text("HC:"+text);
        });
    });
}

$(document).ready(function() {
    getHit();
}
);
```

Client-Server communications

- Whole page reload
  - forms (which can be built on the fly in JS)
- JS based
  - fetch
- Continuing Needs
  - Polling
  - Long Polling
  - EventListeners
  - Node.js

The cost of communicating

- If you load a page HTTPS then all fetch must be HTTPS
  - HTTPS is more expensive
    - Latency increase by switching to HTTPS : the initial SSL handshake (green) requires two (extra) roundtrips before the connection is established, compared to just the one roundtrip required (blue) to establish a TCP connection to the plain unencrypted HTTP port..
    - Bandwidth Increase : The used bandwidth will increase slightly as the header size will increase by a number of bytes for protocol reasons and the effective payload will decrease a due to the framing overhead, and some ciphers will use padding as well. (max 6-7% increase in bandwidth).
    - CPU Load : The most computational expensive part is the public key exchange, after which a relatively efficient symmetric cypher is used.



Assumes that each transmission takes 28ms

Everything done on server or client takes oms

Then http requires at least 112 ms

https requires 224ms

Given that communicating is expensive

Doing more with hit count

- Does client really need to send an ID?
  - $_SERVER[‘HTTP_REFERER']
    - only problem here is that not all clients send it.
    - in practice almost all do
    - Rather than an ID could have JS send this
      - window.location.href
        - Why not just use this as the ID?
- What can I collect about page requesters?
  - What would be intertesting to collect?
    - what infomation about the client is even available?
      - $_SERVER[‘REMOTE_ADDR'];
        - Given IP can reverse DNS and infer origin location ….

Polling

- Easy
  - Just use setInterval in JS
  - PHP looks otherwise unchanged
- But
  - Lots of useless polls
  - May not get information updates quickly

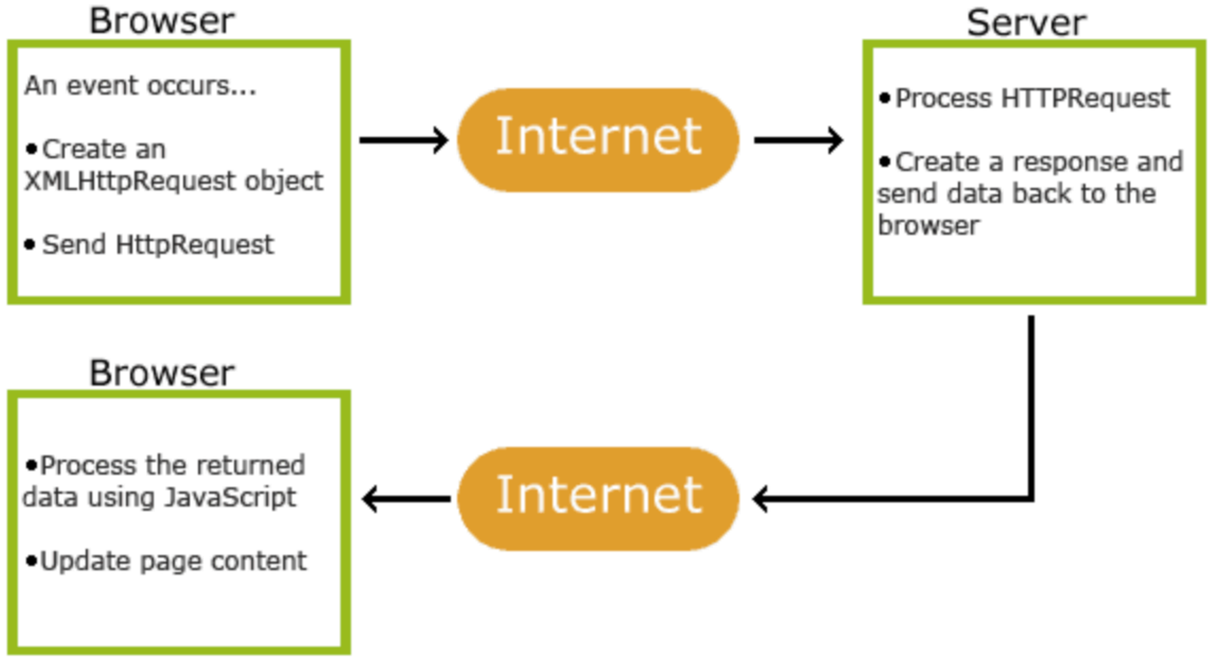
file: polling.html, polling.php

Note 3 different syntaxes for same operation

AJAX

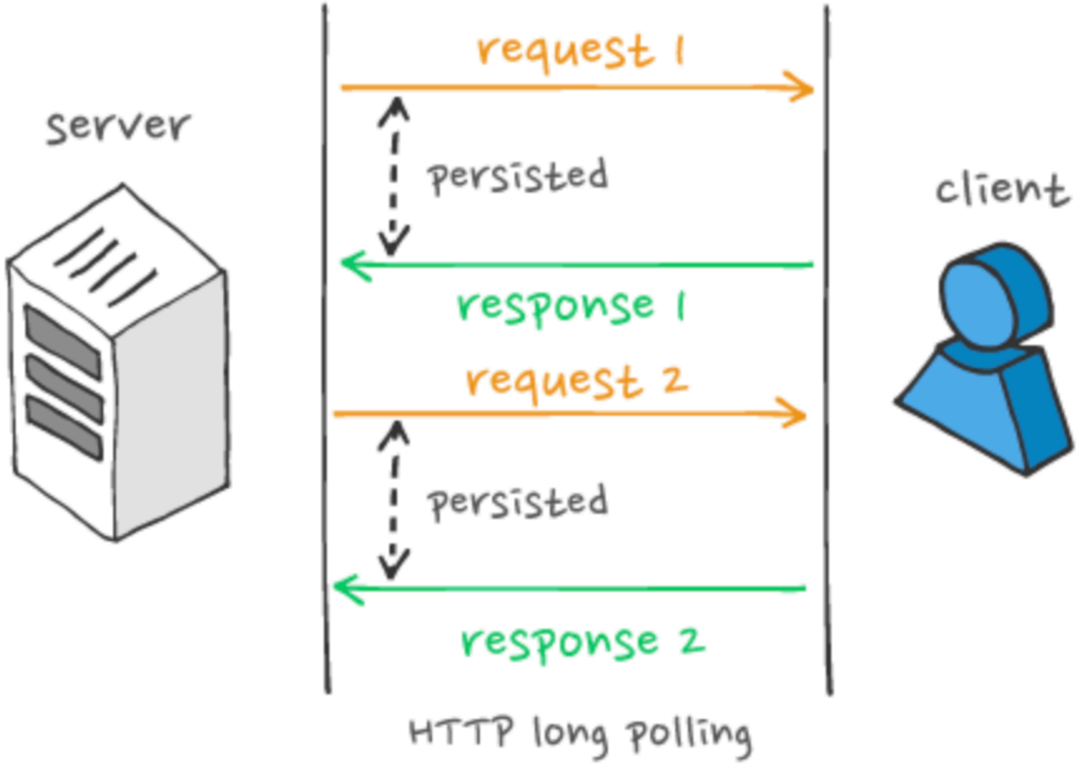
*A*synchronous *J*avaScript *A*nd *X*ML.

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript



Long Polling

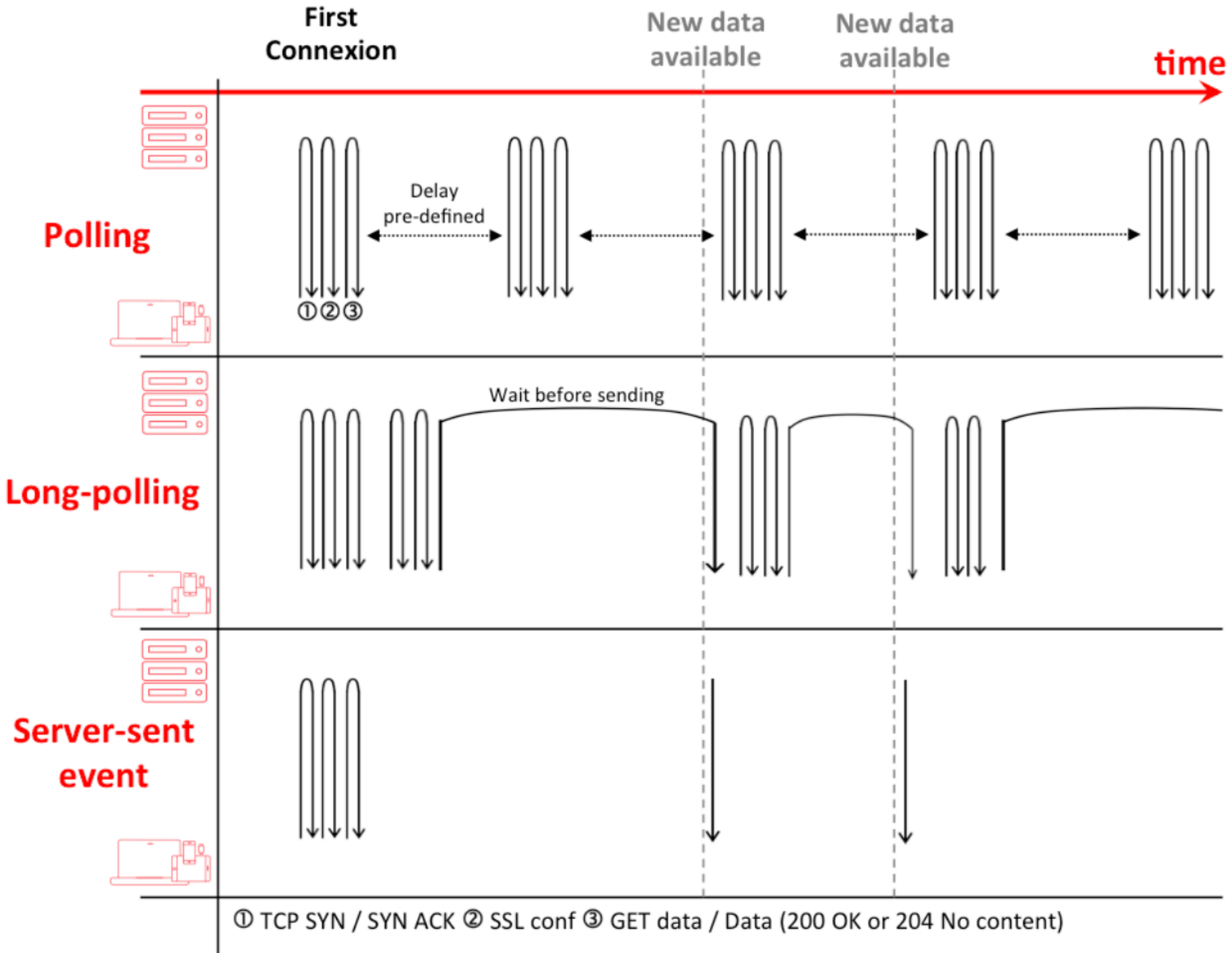
- Core idea
  - Do all the work to set up a connection
  - Do not send back information from server until the something interesting happens
- Advantage:
  - no awkward polling
  - immediate notification when event occurs
  - simple
- Disadvantage:
  - The server architecture must be able to work with many pending connections.
    - Some servers run one process per connection; resulting in as many processes as there are connections,. Each process may consumes a lot of memory.
  - Requires server side code support



files: longpoller.html, longpolled.php

Server Sent Events

- Long Polling without setup/teardown of connection
- Same server-side concerns for open connections
- ONE WAY!!
  - Only time client says anything to server is at setup
  - GET only



Server-side events

- javascript EventSource
  - need to write “handlers” for message types
  - default handlers
- PHP
  - message syntax is important but otherwise can look a lot like polling
- “retry” is a poll-like thing

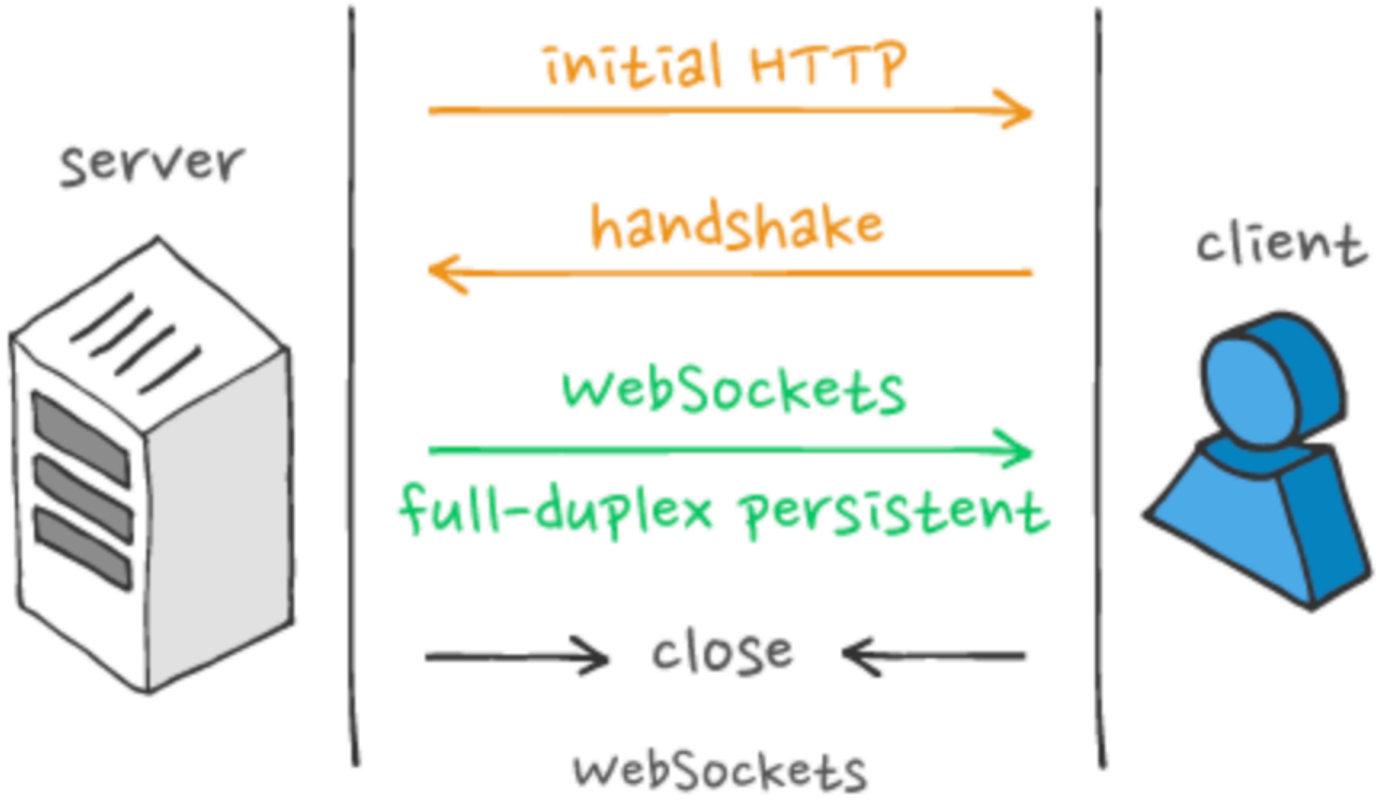
File: eventer.html

evented.php

Fully symmetric communication

Web Sockets

- Node is most widespread implementation
  - Usually called Node.js
- On server side, NOT apache, or Nginx, ..
- low-latency, full-duplex communication makes the location of code less important



For much more: https://www.youtube.com/watch?v=jo_B4LTHi3I

Systems

Towers of Hanoi

- Goal: provide a way for 206 students to have friendly competition with the towers of Hanoi
- Design: a stopwatch with Hanoi instructions
- A backend to generate a graph of times.
  - Graph drawn on html canvas object

Files: towers.html, aaa.php

```
create database if not exists hanoi;

use hanoi;
drop table if exists timedata;

create table timedata (
       id int NOT NULL auto_increment primary
       actor varchar(64),
       witness varchar(5),
       time varchar(10)
       );
```