

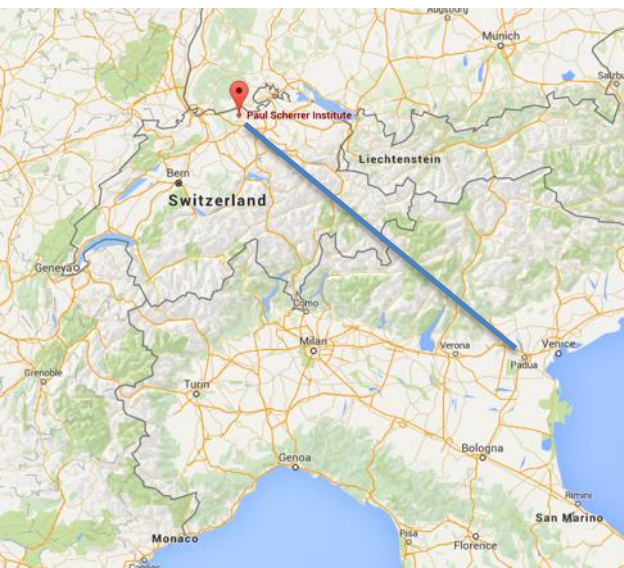
Stefan Ritt, Paul Scherrer Institute, Switzerland

REAL-TIME DATA VISUALIZATION AND CONTROL USING MODERN WEB TECHNOLOGIES

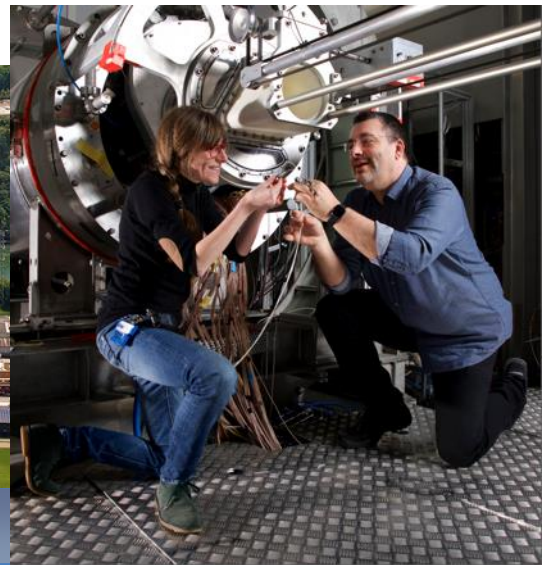
2ND REAL TIME SCHOOL, HO CHI MINH CITY, VIETNAM, 2016

Something about myself

- Studied at University of Karlsruhe, Germany
- Head of muon physics group at the Paul Scherrer Institute, Switzerland
- Developer of
ELOG electronic logbook (midas.psi.ch/eelog)
MIDAS DAQ system (midas.triumf.ca)
DRS chip (www.psi.ch/drs)



Real Time

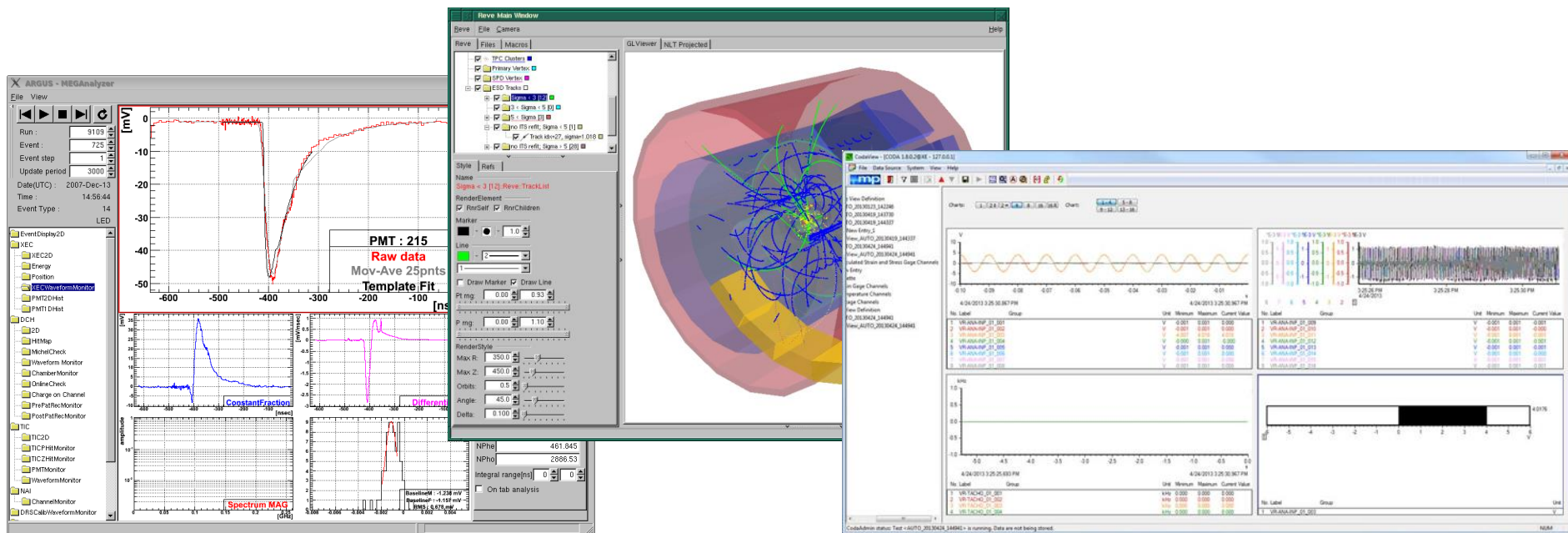


About this Lecture

- Web technologies from a user's point of view (will cover ~5% of HTML) – mainly examples driven
- Slow start with basics, advanced topics at the end
- Log in to wireless network:
SSID **XXX** Password **XXX**
- Please follow all practical examples with your own laptop:
jsfiddle.net/user/stefanritt
- Slides, programs, ... on Dropbox:
db.tt/WWkBEhMQ
- Please ask questions any time

Experimental Data Visualization and Control

- The traditional way
 - Dedicated programs (ROOT, Qt, TCL/TK, ...)
 - Must be compiled for different OS
 - Require certain libraries to be installed
 - Limited smartphone support



A new opportunity

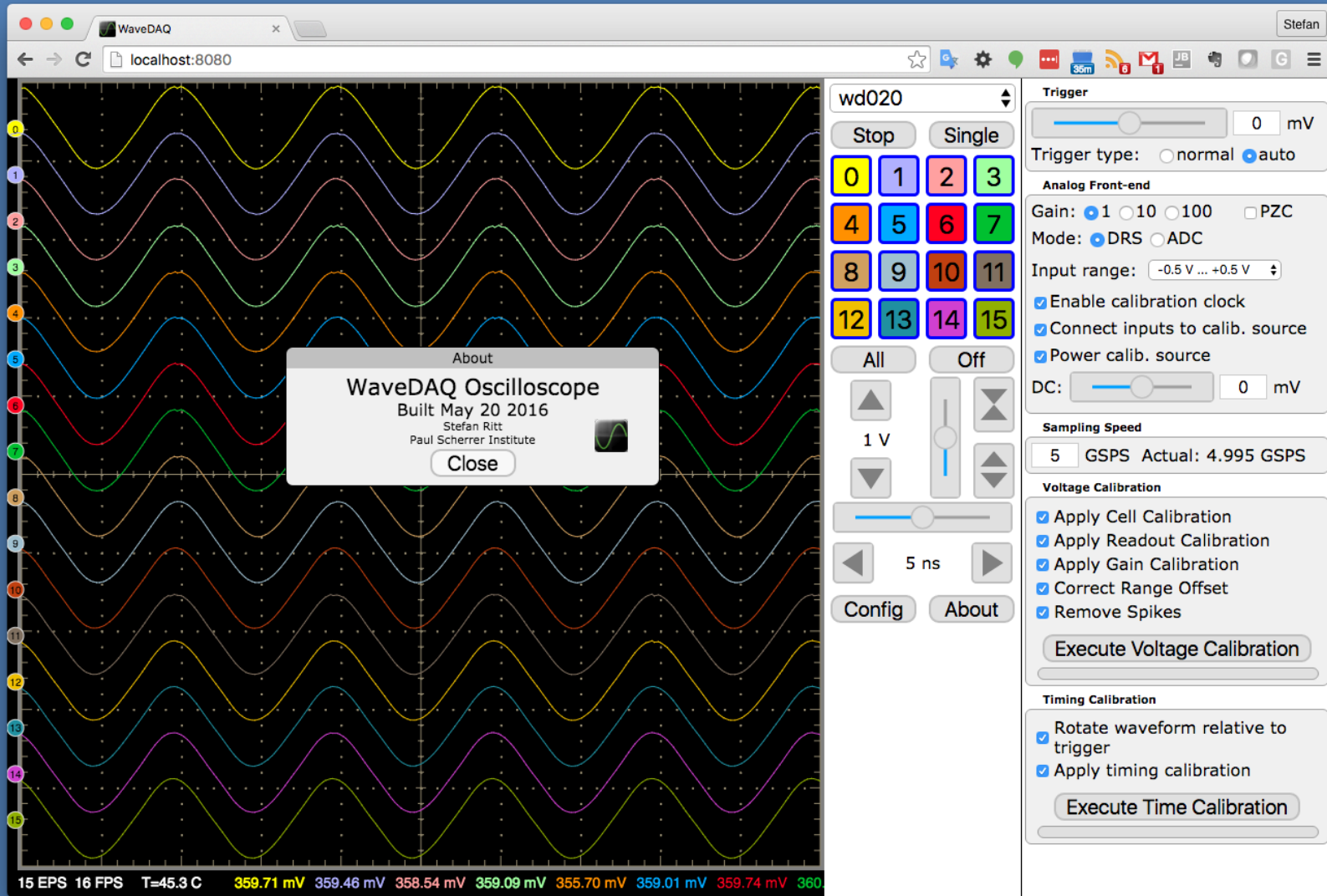


HTML5 – CSS3 – JavaScript – JSON

Advantages of Web Control

- You only need a browser to access your experiment (from home, public terminal, smartphone)
- Software updates get deployed automatically
- Easy creation of web pages showing several experiments
- Modern browsers run JavaScript at the speed of native programs some years ago
- Debugging of code inside the browser

What you will learn – a little teaser



Agenda

- Basic Web technologies
 - HTML
 - CSS
 - Forms
 - JavaScript
 - DOM
- Advanced Web technologies
 - AJAX
 - JSON
 - Canvas

Agenda

- Server side implementation
 - Mongoose server
 - JSON-RPC
 - Implementation on Raspberry Pi
- Advanced web concepts
 - Push technology with Websockets
 - Floating dialog boxes
 - Dialog widgets

Agenda

- Applications
 - Temperature display
 - Turning LEDs on and off on server
 - Pushing information to browsers
 - Oscilloscope application
 - Other Applications

HTML + CSS

Hypertext Markup Language

```
<div>
Hello Real Time Conference
</div>
```

Cascading Style Sheets

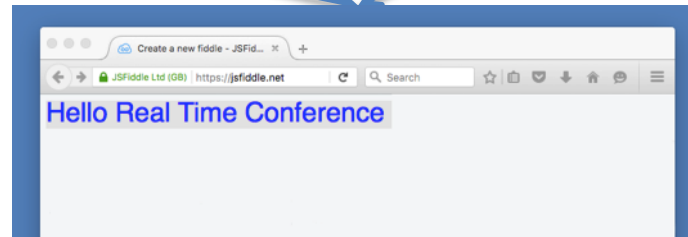
Selector
Declaration
Declaration
Declaration
Declaration

```
div {
  font-family:sans-serif;
  font-size:200%;
  color:blue;
  background-color:#E0E0E0;
}
```

Property Value

Browser

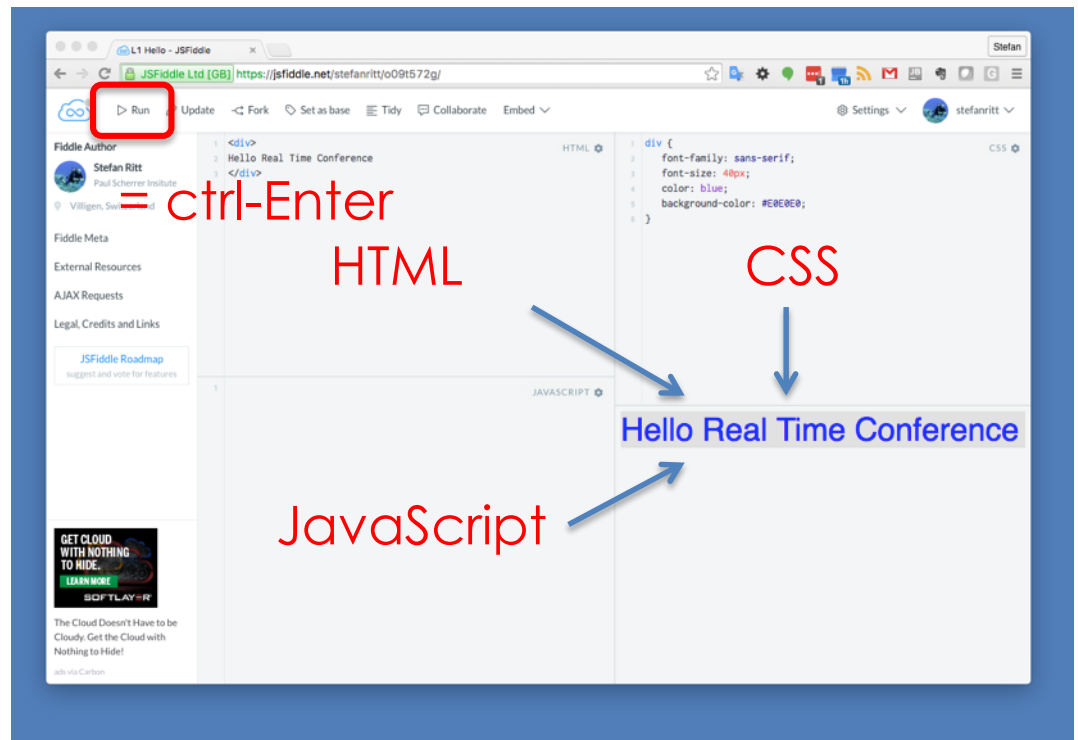
```
<style>
  div {
    color:blue;
    font-size:200%;
  }
</style>
```



```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

jsfiddle.net

- Online tool to test HTML, CSS & JavaScript
- Documentation: doc.jsfiddle.net
- Alternatives:
 - cssdeck.com
 - jsbin.com
 - dabblet.com



First Example

The screenshot shows a JSFiddle editor interface. The top navigation bar includes a search bar, a notification "Your fiddle has an unsaved draft", and a user profile for "stefanritt". The editor is divided into three main sections: HTML, CSS, and JavaScript. The HTML section contains three paragraphs of text with inline styles and classes. The CSS section contains styles for the body, a span, a class named ".subtle", and an ID named "#next". The JavaScript section is currently empty. The preview area on the right shows the rendered output of the code, which is a light gray background with three lines of text: "Welcome to the Real Time Conference.", "Do you enjoy the Real Time tutorial?", and "Will you come to the next Real Time tutorial?". The text is styled according to the CSS rules, with "Real Time" in red, "Real Time" in red and italicized, and "Real Time" in red and italicized.

```
<p>
  Welcome to the <span>Real Time</span> Conference.
</p>
<p>
  Do you enjoy the <span class="subtle">Real Time</span> tutorial?
</p>
<p>
  Will you come to the next <span class="subtle" id="next">Real Time</span>
  tutorial?
</p>
```

```
body {
  font-family: sans-serif;
  font-size:180%;
  color: blue;
  background-color: #E0E0E0;
}

span {
  color: red;
}

.subtle {
  font-style: italic;
}

#next {
  color: black;
}
```

Welcome to the **Real Time** Conference.
Do you enjoy the **Real Time** tutorial?
Will you come to the next **Real Time** tutorial?

Debugging CSS

The screenshot shows a JSFiddle editor with the following HTML and CSS:

```
HTML
1 <p>
2   Welcome to the <span>Real Time</span> Conference.
3 </p>
4
5 <p>
6   Do you enjoy the <span class="subtle">Real Time</span> tutorial?
7 </p>
8
9 <p>
10  Will you come to the next <span class="subtle" id="next">Real Time</span>
11  tutorial?
12 </p>
13
```

```
CSS
1 body {
2   font-family: sans-serif;
3   font-size: 180%;
4   color: blue;
5   background-color: #E0E0E0;
6 }
7
8 span {
9   color: red;
10 }
```

The browser preview shows the rendered output:

Welcome to the **Real Time** Conference.

Do you enjoy the *Real Time* tutorial?

Will you come to the next **Real Time** tutorial?

The CSS Rules pane shows the following styles:

- element { inline}
- #next { inline: 17; color: black; } (indicated by a red arrow)
- .subtle { inline: 13; font-style: italic; } (indicated by a red arrow)
- span { inline: 9; color: red; } (indicated by a red arrow)
- Inherited from body
- body { inline: 2; font-family: sans-serif; font-size: 180%; color: blue; } (indicated by a red arrow)

CSS Selectors

Selector	Example	Description
.	.subtle	Selects all elements with class="subtle"
#	#next	Selects all elements with id="next"
<i>element</i>	p	Selects all elements
<i>element > element</i>	div > p	Selects all <p> elements where the parent is a <div> element
<i>element[attribute]</i>	input[type="text"]	Select all <input type="text"> elements
<i>element:nth-child(i)</i>	p:nth-child(2)	Select all <p> elements which are the second child of their parent

```
p:nth-child(odd) {  
    background: #F0F0F0;  
}  
p:nth-child(even) {  
    background: #C0C0C0;  
}
```

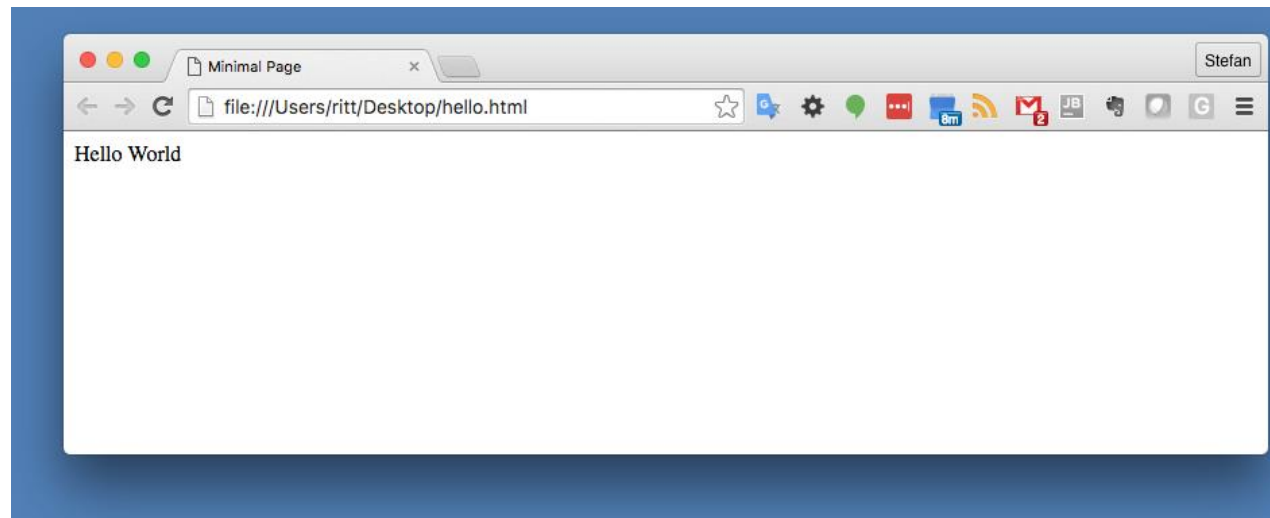
http://www.w3schools.com/cssref/css_selectors.asp

HTML - CSS

- Do make use of separated HTML and CSS `Hello Conference` is bad practice!
- Overview of CSS options:
<http://www.w3schools.com/css/>
- Debug CSS e.g. with Chrome Develop Tools or Firefox Inspector
- Google Search is your friend:
“CSS text size” → { font-size: 200%; }
- www.w3schools.com, wiki.selfhtml.org (only German)

Full minimal HTML5

```
<!doctype html> ← Tells the browser that this is an HTML5 file
<html lang="en"> ← Primary language
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"> ← Encoding
    <title>Minimal Page</title>
    <link rel="stylesheet" type="text/css" href="style.css"> ← Optional
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    Hello World ← Page contents
  </body>
</html>
```



Forms

- Forms must be embedded in `<form>... </form>` tag
- Text input, check box, radio box, drop-down list, submit button
- Form contents gets submitted in URL (for “get” method):
`http://your.host?name=Stefan&gender=Male`
- Form submission is obsolete!

Form Example

The screenshot displays the JSFiddle interface for a project titled "L3 Form". The browser address bar shows the URL "https://jsfiddle.net/stefanritt/f5h7x5vf/". The interface includes a top navigation bar with options like "Run", "Update", "Fork", "Set as base", "Tidy", "Collaborate", and "Embed". On the right, there are "Settings" and a user profile for "stefanritt".

The main workspace is divided into three panels:

- HTML:** Contains the following code:

```
1 <form>
2 <p>
3   Hometown:
4   <input type="text" name="hometown" size="30">
5 </p>
6
7 <p>
8   Gender:
9   <input type="radio" name="gender" value="Male">Male
10  <input type="radio" name="gender" value="Female">Female
11 </p>
12
13 <p>
14   Age:
15   <select name="age">
16     <option value="0">0-20</option>
17     <option value="20">20-40</option>
18     <option value="40">40-60</option>
19     <option value="60">60-80+</option>
20   </select>
21 </p>
22
23 <p>
24   <textarea name="comment" rows="10" cols="20">Enter comment here
25   </textarea>
26
27 </p>
28   <input type="submit" value="Submit">
29 </form>
30
```
- CSS:** Contains the following code:

```
1 body {
2   font-family: sans-serif;
3 }
4
```
- JAVASCRIPT:** This panel is currently empty.

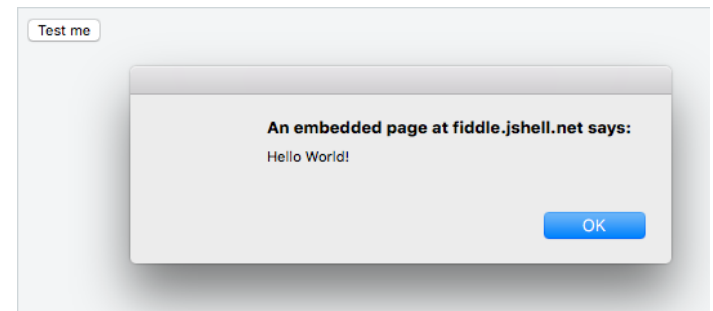
The right side of the interface shows a live preview of the rendered form. It includes a text input field for "Hometown:", radio buttons for "Gender: Male" and "Gender: Female", a dropdown menu for "Age:" with "0-20" selected, a text area for "Enter comment here", and a "Submit" button.

JavaScript

- Standardized *interpreted* language running in your browser

- Very slim:

```
<input type="button" value="Test me" onclick="alert('Hello World')">
```



- Similar to C/C++
- Browser-"war" for the fastest interpreter: JavaScript runs as fast as compiled code about ten years ago
→ JS can be used for complicated things

JavaScript Example

The screenshot shows a JSFiddle browser window with the following content:

Browser Address Bar: L4 JavaScript - JSFiddle | https://jsfiddle.net/stefanritt/eh40hgo4/

Navigation Bar: Run, Update, Fork, Set as base, Tidy, Collaborate, Embed

Left Panel (Fiddle Meta): L4 JavaScript
Simple JavaScript program to add two number

HTML Editor:

```
1 <p>
2   X:
3   <input type="text" id="x">
4 </p>
5
6 <p>
7   Y:
8   <input type="text" id="y">
9 </p>
10
11 <p>
12   <span id="result">Result:</span>
13 </p>
14
15 <p>
16 <input type="button" onClick="add()" value="Add">
17 </p>
18
```

JAVASCRIPT Editor:

```
1 function add() {
2   var x = document.getElementById("x").value;
3   var y = document.getElementById("y").value;
4   var sum = parseInt(x) + parseInt(y);
5   alert(sum);
6
7   document.getElementById("result").innerHTML = "Result: " + sum;
8 }
9
```

Preview:

X:

Y:

Result: 3

Debugging JavaScript

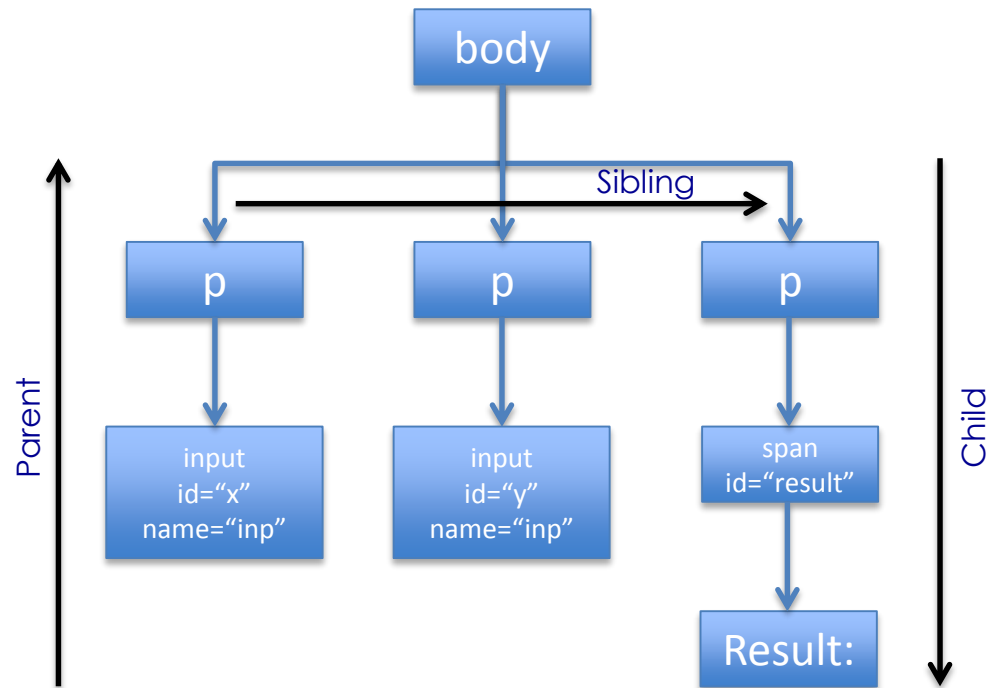
The screenshot displays a JSFiddle browser window with the following components:

- Browser Address Bar:** Shows the URL `https://jsfiddle.net/stefanritt/eh40hgo4/`.
- Debugger Interface:** A "Paused in debugger" banner is visible at the top of the code editor.
- Code Editor:** Contains HTML, CSS, and JavaScript code. The JavaScript code defines an `add()` function that takes two inputs, calculates their sum, and updates the page content. The current execution is paused at line 49: `var x = document.getElementById("x").value;`.
- DOM Preview:** Shows the rendered page with two input fields labeled "X:" and "Y:" containing the values "1" and "2" respectively, and a "Result: 3" label with an "Add" button.
- Developer Tools:** The "Sources" panel is open, showing the `osc.js` file. The "Call Stack" panel shows the `add` function at index 54 and the `onclick` event at index 80. The "Scope" panel shows local variables: `sum: 3`, `this: Window`, `x: "1"`, and `y: "2"`.
- Console:** Shows the current state of the page with the "Result: 3" output.

DOM

Document Object Model behind any web page *plus* interface to that model

```
<body>
  <p>
    <input id="x" name="inp">
  </p>
  <p>
    <input id="y" name="inp">
  </p>
  <p>
    <span id="result">
      Result:
    </span>
  </p>
</body>
```



JavaScript DOM modifications

- Access by ID:
`var e = document.getElementById("result");`
- Access by Name:
`var e = document.getElementsByName("input");`
`for (i=0 ; i<e.length ; i++)`
`e[i].value = "0";`
- Access by Class Name:
`document.getElementsByClassName("subtle");`
- By relation:
`e.firstChild e.nextSibling e.parentNode ...`
- Predefined elements:
`document.title document.images ...`
- CSS Style:
`e.style.color e.style.fontFamily ...`

jQuery

- jQuery is a JavaScript library which simplifies JavaScript programming
- Include it with

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js">
```
- Use "\$" to select elements and special jQuery functions

JavaScript

```
var e = document.getElementsByClassName("subtle");
for (var i=0 ; i<e.length ; i++)
    e.style.color = "red";
```

jQuery

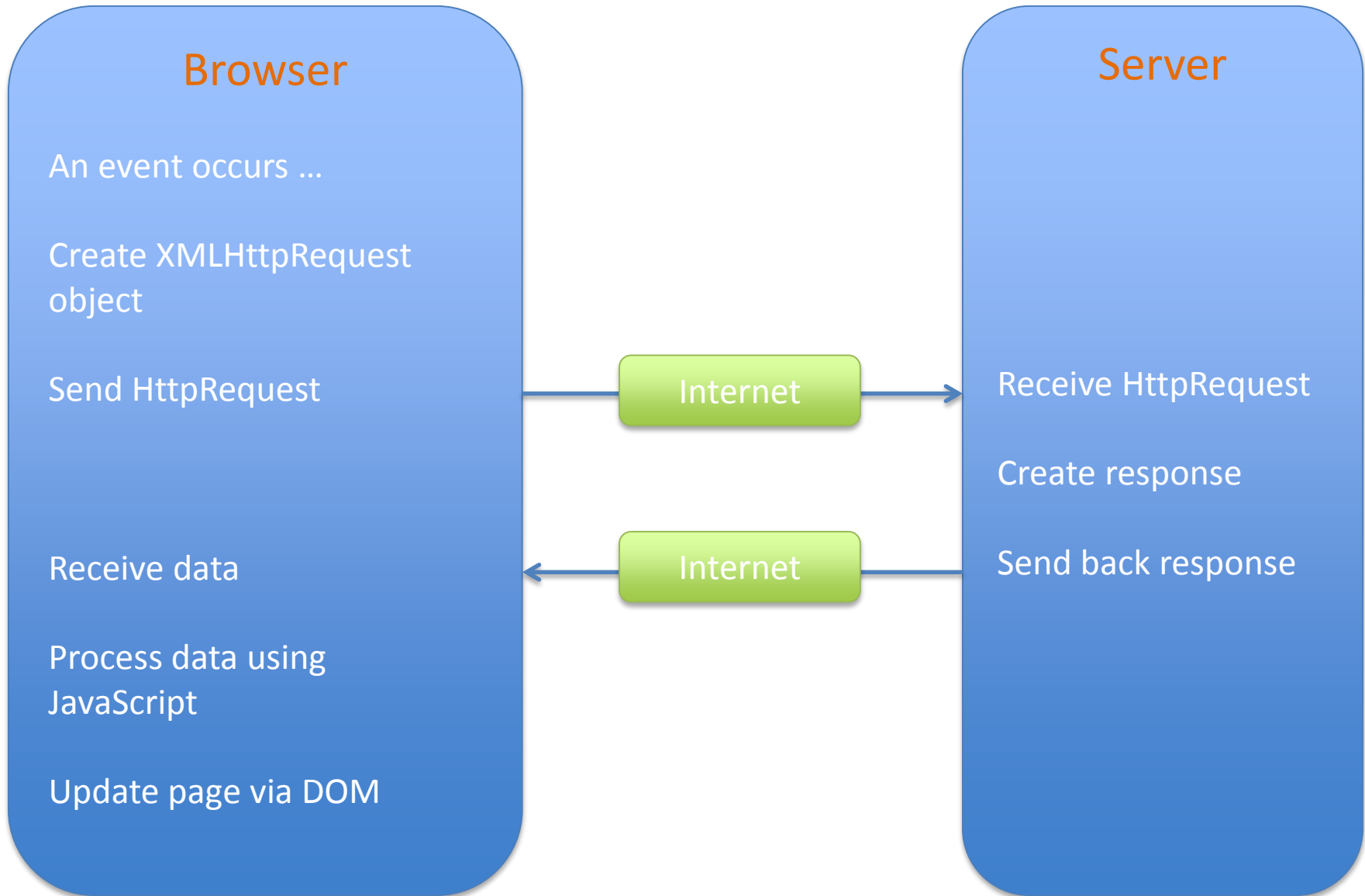
```
$(".subtle").css("color", "red");
```

CSS selector 

AJAX

- Old days: Form – Submit – New Page
- Slow, network intensive, page flicker
- **A**synchronous **J**avascript and **X**ML
 - Update web page without reloading the whole page
 - Send data to the server AFTER the page has loaded
 - Receive data from server AFTER the page has loaded
 - Technique for fast dynamic web pages
- Getting popular in 2005 by Google Suggest

How AJAX Works



AJAX example

The screenshot shows the JSFiddle interface for a project titled "L5 Ajax - JSFiddle". The browser address bar shows the URL "https://jsfiddle.net/stefanritt/x71nud5v/". The interface includes a toolbar with options like Run, Update, Fork, Set as base, Tidy, Collaborate, and Embed. The left sidebar shows the Fiddle Author (stefanritt) and the title "L5 Ajax". The main area is divided into three panels: HTML, CSS, and JAVASCRIPT. The HTML panel contains the following code:

```
1 <div id="result">
2   Idle
3 </div>
4
5 <p>
6   <input type="button" value="Send" onclick="load()">
7 </p>
8
```

The CSS panel contains the following code:

```
1 body {
2   font-family: sans-serif;
3 }
4
```

The JAVASCRIPT panel contains the following code:

```
1 function load() {
2   var xhttp = new XMLHttpRequest();
3   xhttp.onreadystatechange = function() {
4     if (xhttp.readyState == 4 && xhttp.status == 200) {
5       document.getElementById("result").innerHTML = xhttp.responseText;
6     }
7   };
8   xhttp.open("POST", "/echo/html/", true);
9   xhttp.send("html=Response received&delay=3");
10 }
11
```

The live preview area shows the rendered output, which is a button labeled "Send". The status above the preview is "Idle".

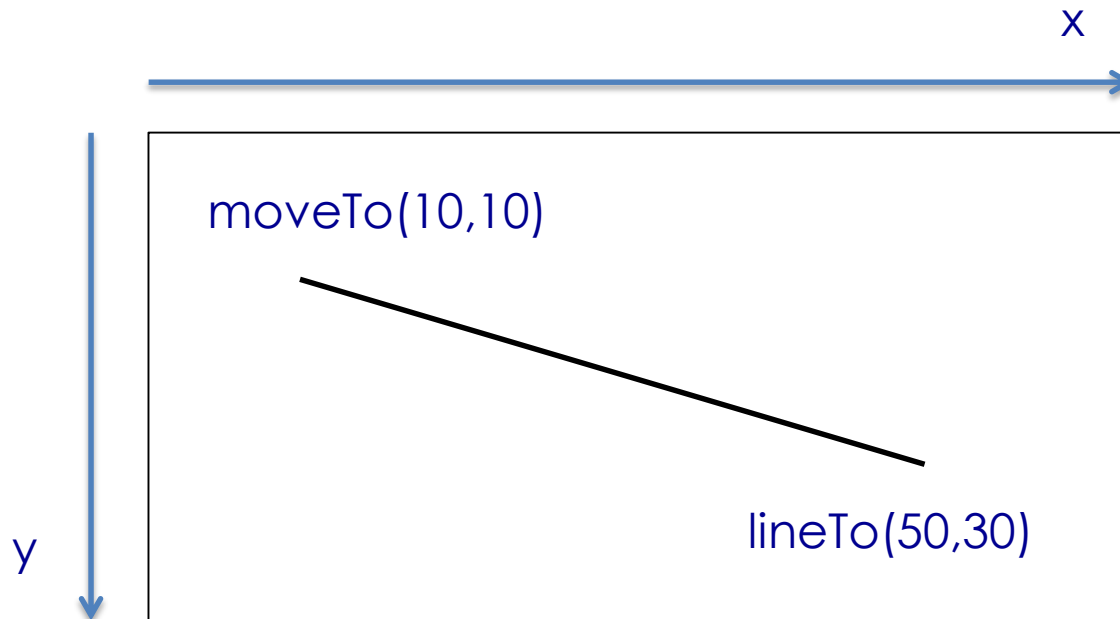
JSON

- JavaScript Object Notation
- ASCII format for storing and transportation of objects
- Pure text format, language independent, self-describing

JSON	Notation	Example
Data	"name": "value"	"firstName": "Stefan"
Object	{ Data, Data }	{ "firstName": "Stefan", "lastName": "Ritt" }
Array	"name": [Object, Object]	"attendees": [{ "firstName": "Stefan", "lastName": "Ritt" }, { "firstName": "Linus", "lastName": "Torvalds" }]

Canvas

- Basic HTML is mainly for text display
- Vector graphics with SVG, icons and **canvas**
- A canvas is a HTML5 element on which one can draw dynamic graphics via JavaScript



Canvas Example

L7 Canvas - JSFiddle

JSFiddle Ltd (GB) | https://jsfiddle.net/stefanritt/0rj05pqq/

Run Update Fork Set as base Tidy Collaborate Embed

Settings stefanritt

Fiddle Author

Fiddle Meta

L7 Canvas

Simple Canvas Demo

```
1 <h1>
2 Canvas Demo
3 </h1>
4
5 <canvas id="cvs" width="200" height="200"></canvas>
6
```

HTML

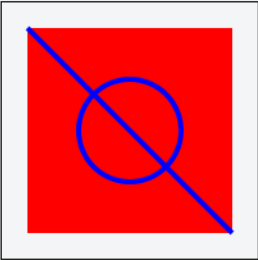
```
1 body {
2   font-family: sans-serif;
3 }
4
5 canvas {
6   border: 1px solid black;
7 }
8
```

CSS

```
1 var c = document.getElementById("cvs");
2 var ctx = c.getContext("2d");
3
4 ctx.fillStyle = "red";
5 ctx.fillRect(20, 20, 160, 160);
6
7 ctx.beginPath();
8 ctx.lineWidth = 4;
9 ctx.strokeStyle = "blue";
10 ctx.moveTo(20, 20);
11 ctx.lineTo(180, 180);
12 ctx.stroke();
13
14 ctx.beginPath();
15 ctx.arc(100, 100, 40, 0, 2 * Math.PI);
16 ctx.stroke();
17
```

JAVASCRIPT

Canvas Demo



The image shows a canvas with a red square background. A blue circle is drawn in the center of the square, and a blue diagonal line is drawn from the top-left corner to the bottom-right corner of the square. The entire canvas is enclosed in a black border.

More complex example

The screenshot shows a JSFiddle editor window titled "L8 Marbles - JSFiddle". The browser address bar shows the URL "https://jsfiddle.net/stefanritt/66q2bmve/". The editor interface includes a top navigation bar with buttons for "Run", "Update", "Fork", "Set as base", "Tidy", "Collaborate", and "Embed". On the right side of the top bar, there are "Settings" and a user profile for "stefanritt".

The editor is divided into three main sections:

- HTML:** Contains a single line of code: `<canvas id="cvs"></canvas>`.
- CSS:** Contains a single rule: `body { margin: 0; }`.
- JAVASCRIPT:** Contains the following code:

```
1 var marbles = [];  
2  
3 resizeCanvas();  
4 window.addEventListener("resize", resizeCanvas);  
5 window.addEventListener("click", newMarble);  
6 window.setTimeout(redraw, 10);  
7  
8 var marbles = new Array();  
9  
10 function resizeCanvas() {  
11     var c = document.getElementById("cvs");  
12     c.width = window.innerWidth;  
13     c.height = window.innerHeight;  
14 }  
15  
16 function newMarble(event) {  
17     marbles.push(new marble(event.clientX, event.clientY));  
18 }  
19  
20 function redraw() {  
21     var ctx = document.getElementById("cvs").getContext("2d");  
22  
23     ctx.fillStyle = "#404040";  
24     ctx.fillRect(0, 0, ctx.canvas.width, ctx.canvas.height);  
25  
26     for (var i = 0; i < marbles.length; i++) {  
27         var x = marbles[i].x;  
28         var y = marbles[i].y;  
29  
30         var grd = ctx.createRadialGradient(x, y, 1, x, y, 15);  
31         grd.addColorStop(0, "white");  
32         grd.addColorStop(1, "black");
```

The left sidebar shows the "Fiddle Author" as "L8 Marbles" and "Fiddle Meta" as "No description".

Balls 1

```
<head>
<script>
var balls = [];
var lastTime = 0;
var lastTimeFrame = 0;
var nFrames = 0;
var fps = 0;

function init() {
  resizeCanvas();

  // attach event handlers to canvas
  var c = document.getElementById("cvs");
  c.addEventListener("resize", resizeCanvas);
  c.addEventListener("click", newBall);
  c.addEventListener("touchstart", newBall, true);
  window.addEventListener("keypress", keyPress)

  window.requestAnimationFrame(frame);
}

function resizeCanvas()
{
  // stretch canvas to the whole client area
  var c = document.getElementById("cvs");
  c.width = document.documentElement.clientWidth;
  c.height = document.documentElement.clientHeight;
  if (c.height > 1000)
    c.height = 1000;
}
```

```
function newBall(e)
{
  // create new ball at cursor position
  if (e.type == "click")
    balls.push(new ball(e.clientX, e.clientY));
  else if (e.type == "touchstart") {
    e.preventDefault();
    balls.push(new ball(e.targetTouches[0].clientX,
                        e.targetTouches[0].clientY));
  }
}

function keyPress(e)
{
  var c = (typeof e.which == "number") ?
    e.which : e.keyCode;
  if (c == 'c'.charCodeAt(0)) // 'c' clears all balls
    balls = [];
  if (c == ' '.charCodeAt(0)) // space pushes up
    balls.forEach(function(m) {
      m.vy = Math.random() * 1000 - 1500;
      m.vx = Math.random() * 1000 - 500;
    });
  e.preventDefault();
}
```

Balls 2

```
function frame()
{
    var ctx =
document.getElementById("cvs").getContext("2d");

    // fill background
    ctx.fillStyle = "#606060";
    ctx.fillRect(0, 0, ctx.canvas.width,
ctx.canvas.height);

    // time since last frame
    var now = new Date().getTime(); // get time in ms
    var dt = lastTime > 0 ?
        (now - lastTime) / 1000 : 0;
    lastTime = now;

    // calculate frames per second
    nFrames++;
    if (now >= lastTimeFrame + 1000) {
        lastTimeFrame = now;
        fps = nFrames;
        nFrames = 0;
    }

    // display number of frames
    ctx.font = "20px sans-serif";
    ctx.fillStyle = "white";
    ctx.strokeStyle = "none";
    ctx.fillText(fps, 10, 20);
```

```
for (var i = 0; i < balls.length; i++) {
    var x = balls[i].x;
    var y = balls[i].y;

    var grd = ctx.createRadialGradient(x, y,
        1, x, y, 15);
    grd.addColorStop(0, "white");
    grd.addColorStop(1, "black");
    ctx.fillStyle = grd;

    // draw ball
    ctx.beginPath();
    ctx.arc(x, y, 10, 0, 2 * Math.PI);
    ctx.fill();

    // integrate equations of motion
    balls[i].x += balls[i].vx * dt;
    balls[i].y += balls[i].vy * dt;
```

Balls 3

```
// handle wall reflections
if (balls[i].x + balls[i].vx*dt <= 10) {
  balls[i].x = 10;
  balls[i].vx = -balls[i].vx;
}
if (balls[i].x + balls[i].vx*dt >=
  ctx.canvas.width - 10) {
  balls[i].x = ctx.canvas.width - 10;
  balls[i].vx = -balls[i].vx;
}
if (balls[i].y + balls[i].vy*dt <= 10) {
  balls[i].vy = -balls[i].vy;
  balls[i].y = 10;
}
if (balls[i].y + balls[i].vy*dt >=
  ctx.canvas.height - 10) {
  balls[i].y = ctx.canvas.height - 10;
  balls[i].vy = -balls[i].vy;
}

// apply friction and gravity
balls[i].vx *= 0.995; // friction
balls[i].vy *= 0.995;
balls[i].vy += 3;    // gravity
}

window.requestAnimationFrame(frame);
}
```

```
function ball(x, y)
{
  // initialize new ball with random velocity
  this.x = x;
  this.y = y;
  this.vx = Math.random() * 1000 - 500;
  this.vy = Math.random() * 1000 - 1000;
}
</script>

<style>
body { margin: 0; }
</style>
</head>

<body onload="init()">
<canvas id="cvs"></canvas>
</body>
```

Dialog Box

L10 Dialog Box - JSFiddle

JSFiddle Ltd (GB) | https://jsfiddle.net/stefantritt/eeLn6h6y/

Run Update Fork Set as base Tidy Collaborate Embed

Settings stefantritt

Fiddle Author

Fiddle Meta

L10 Dialog Box

Simple dialog box which is draggable over the background text

```
HTML
1 <div>
2   <p>
3     <input type="button" value="Dialog" onClick="showDialog();">
4   </p>
5   <p id="bg">Background</p>
6 </div>
7
8 <div id="dialog" class="dialog">
9   <div class="dialogTitle" id="dialogTitle">Dialog Title</div>
10  <div class="dialogContents">
11    Dialog contents
12    Dialog contents
13    Dialog contents
14  </div>
15  <input type="button" value="close" />
16 </div>
17
18
19
20
21
22
23
24
JAVASCRIPT
1 function showDialog() {
2   var e = document.getElementById("dialog");
3   e.style.display = "block";
4   e.style.left = document.documentElement.clientWidth / 2 -
5   e.offsetWidth / 2 + "px";
6   e.style.top = document.documentElement.clientHeight / 2 -
7   e.offsetHeight / 2 + "px";
8
9   window.addEventListener("mousedown", dialogDrag, true);
10  window.addEventListener("mousemove", dialogDrag, true);
11  window.addEventListener("mouseup", dialogDrag, true);
12  window.addEventListener("touchstart", dialogDrag, true);
13  window.addEventListener("touchmove", dialogDrag, true);
14 }
15
16 function hideDialog() {
17   var e = document.getElementById("dialog");
18   e.style.display = "none";
19 }
20
21 var Ax, Ay, Dx, Dy;
22
23 function dialogDrag(e) {
24   var x = undefined;
25   var dlg = document.getElementById("dialog");
26 }
```

```
CSS
1 body {
2   background-color: #E0FFE0;
3 }
4 #bg {
5   font-family: sans-serif;
6   font-size: 96px;
7   color: #A0A0FF;
8 }
9 .dialog {
10  background-color: #E0FFE0;
11  border: 1px solid #ccc;
12  padding: 10px;
13  width: 200px;
14  margin: auto;
15  text-align: center;
16 }
17 .dialogTitle {
18  background-color: #ccc;
19  padding: 5px;
20  margin-bottom: 5px;
21 }
22 .dialogContents {
23  padding: 5px;
24 }
```

Dialog

Dialog Title

Dialog contents
Dialog contents
Dialog contents

close

Browser Compatibility <http://caniuse.com>

Can I use radial-gradient ? [Settings](#)

1 result found

CSS Repeating Gradients - CR

Global 88.71% + 0.01% = 88.72%
unprefixed: 77.74%

Method of defining a repeating linear or radial color gradient as a CSS image.

[Current aligned](#) [Usage relative](#) [Show all](#)

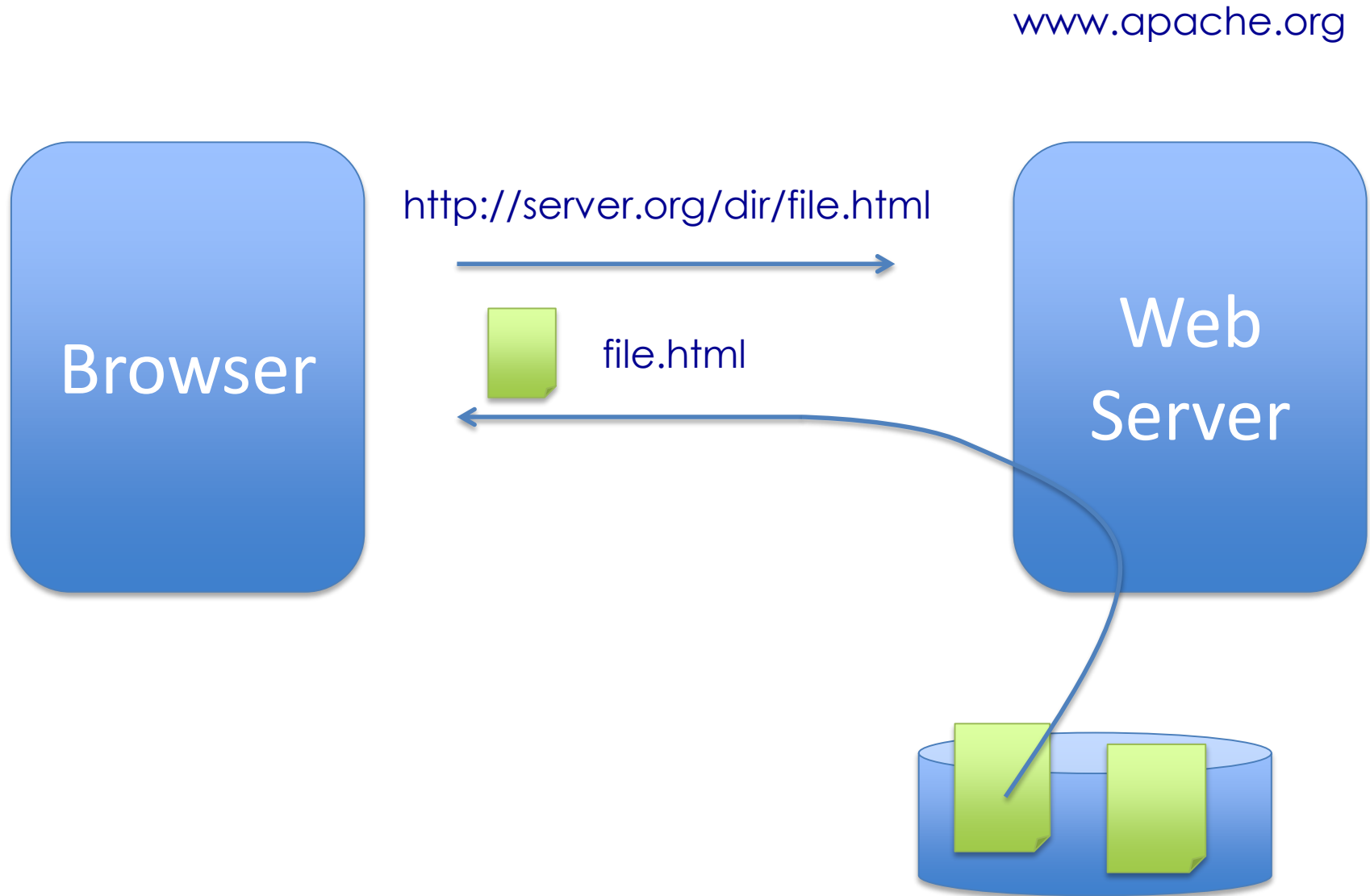
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
			45					4.3	
8			48			8.4		4.4	
9		45	49	9	36	9.2		4.4.4	
11	13	46	50	9.1	37	9.3	8	50	50
	14	47	51	TP	38				
		48	52		39				
		49	53						

[Notes](#) [Known issues \(1\)](#) [Resources \(4\)](#) [Feedback](#)

Firefox 10+, Chrome 26+ and Opera 11.6+ also support the new "to (side)" syntax.

■ = Supported ■ = Not supported ■ = Partial support ■ = Support unknown

Web servers

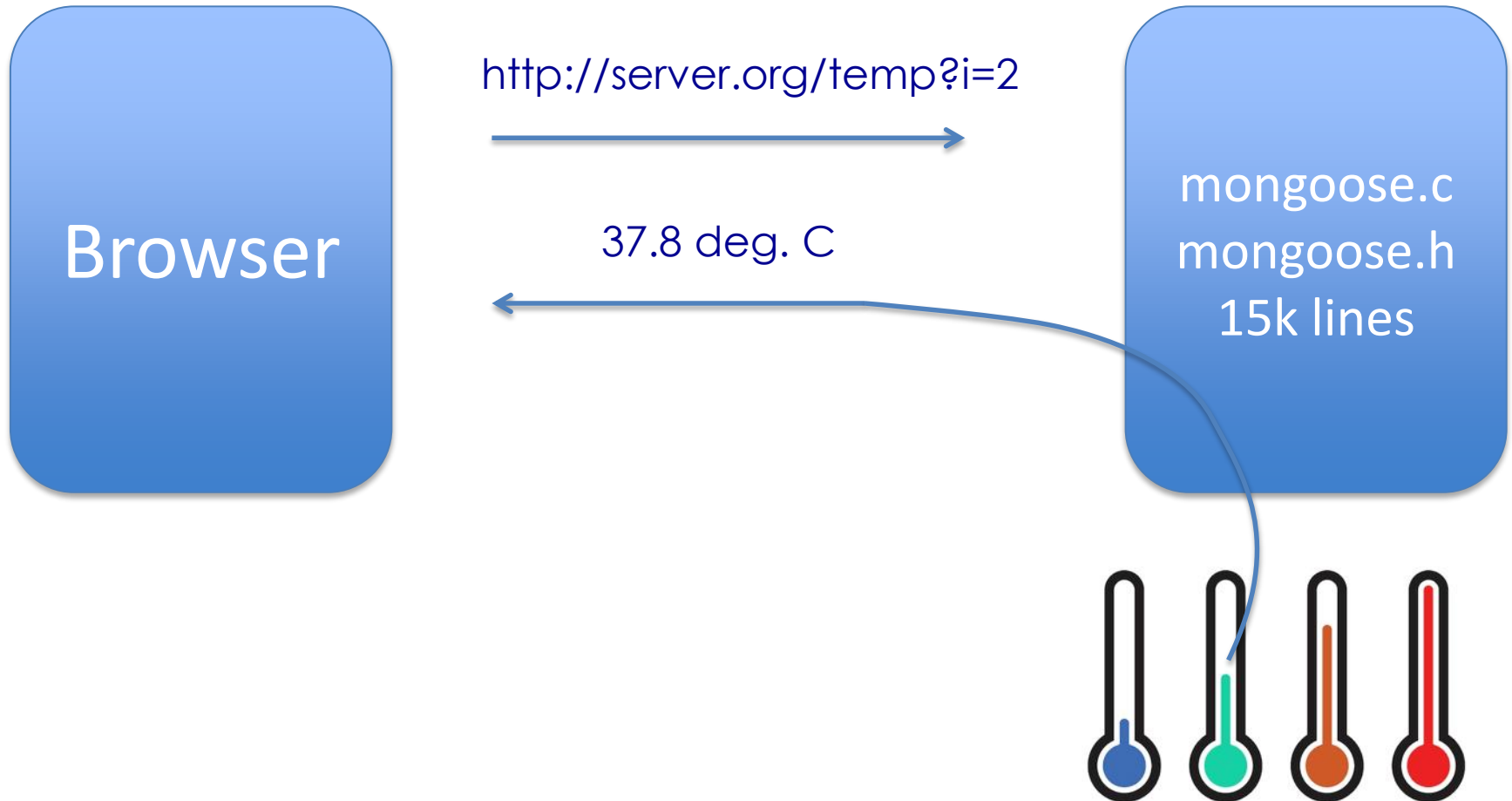


Web Servers

- Web servers are historically file based
- Apache is a large complex but powerful server
- For real-time control and monitoring, we need:
 - Slim server (e.g. embedded devices)
 - Servers with direct access to hardware
 - Support of protocols such as Websockets and SSL
 - Should compile everywhere
 - Easy to install and extend

Mongoose Server

<https://github.com/cesanta/mongoose>



Minimal mongoose program 1

```
#include "mongoose.h"

static struct mg_serve_http_opts s_http_server_opts;
static char *s_http_port = "8080";

int main(int argc, char *argv[])
{
    struct mg_mgr mgr;
    struct mg_connection *nc;

    mg_mgr_init(&mgr, NULL);
    nc = mg_bind(&mgr, s_http_port, ev_handler);
    if (nc == NULL) {
        fprintf(stderr, "Error starting server on port %s\n", s_http_port);
        exit(1);
    }

    mg_set_protocol_http_websocket(nc);
    s_http_server_opts.document_root = ".";
    s_http_server_opts.enable_directory_listing = "yes";

    printf("Starting server on port %s\n", s_http_port);

    while(1) {
        mg_mgr_poll(&mgr, 1000);
    }

    return 0;
}
```

Minimal mongoose program 2

```
static void ev_handler(struct mg_connection *nc, int ev, void *ev_data)
{
    char str[100];
    int index;
    double temp;
    struct http_message *hm = (struct http_message *) ev_data;

    if (ev == MG_EV_HTTP_REQUEST) {
        if (mg_vcmp(&hm->uri, "/temp") == 0) {

            // retrieve index from URL
            mg_get_http_var(&hm->query_string, "index", str, sizeof(str));
            index = atoi(str);

            // read temperature
            temp = 20+(int)(100.0*rand()/RAND_MAX)/10.0;

            // send reply
            mg_printf(nc, "%s", "HTTP/1.1 200 OK\r\nTransfer-Encoding: chunked\r\n\r\n");
            mg_printf_http_chunk(nc, "{\r\n");
            mg_printf_http_chunk(nc, "  \"index\": %d,\r\n", index);
            mg_printf_http_chunk(nc, "  \"temp\": %lg\r\n", temp);
            mg_printf_http_chunk(nc, "}\r\n");
            mg_send_http_chunk(nc, "", 0); // send empty chunk as end of response
        } else {
            // serve static content
            mg_serve_http(nc, hm, s_http_server_opts);
        }
    }
}
```

HTML page

```
<!DOCTYPE html>
<html>
<head>
  <title>Temperature Display</title>

<style>
  body {
    margin: 0;
  }
  #temp {
    height: 300px;
    background: yellow;
    background: linear-gradient(red,
yellow);

    font-family: sans-serif;
    font-size: 250px;
    text-align: center;
  }
</style>
```

```
<script>

window.setTimeout(load, 1000);

function load()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 &&
xhttp.status == 200) {
      o = JSON.parse(xhttp.responseText);
      document.getElementById("temp").innerHTML
= o.temp;
      window.setTimeout(load, 1000);
    }
  };
  xhttp.open("GET", "/temp?index=1", true);
  xhttp.send();
}

</script>
</head>

<body>
  <div id="temp">0.0</div>
</body>

</html>
```

HTTP Requests

- Hypertext Transfer Protocol defined in <https://tools.ietf.org/pdf/rfc2616.pdf>
- For http:// requests, browser opens a TCP connection to port 80 of the server, sends a request header, and receives a response in plain text
- For https:// requests, encryption (SSL) is used to port 443
- Typical request:

```
GET /temp?index=1 HTTP/1.1\r\n
```


Network monitor (Chrome Developer Tools)

The screenshot shows the Chrome Developer Tools Network Monitor. The browser address bar displays `localhost:8080/temp?index=1`. The page content shows a JSON object: `{ "index": 1, "temp": 20 }`. The Network tab is active, showing a list of requests with the selected request `temp?index=1` highlighted. The waterfall chart on the right details the request lifecycle with the following times:

Phase	TIME
Connection Setup	
Queueing	0.62 ms
Stalled	0.46 ms
DNS Lookup	0.18 ms
Initial connection	0.39 ms
Request/Response	
Request sent	0.06 ms
Waiting (TTFB)	0.44 ms
Content Download	0.76 ms
Explanation	2.96 ms

The 'Explanation' bar in the waterfall chart is highlighted in blue. Below the waterfall chart, a table provides details for the selected request:

Name	Method	Status	Type	Initiator	Size	Time
temp?index=1	GET	200	docum...	Other	107 B	2 ms

At the bottom of the Network Monitor, a summary bar displays: `1 / 13 requests | 107 B / 107 B transferred | Finish: 286 ms | DOMContentLoaded: 283 ms | Load: 283 ms`. A blue arrow points from the JSON response area to the 'Explanation' bar in the waterfall chart.

HTTP Request and Response

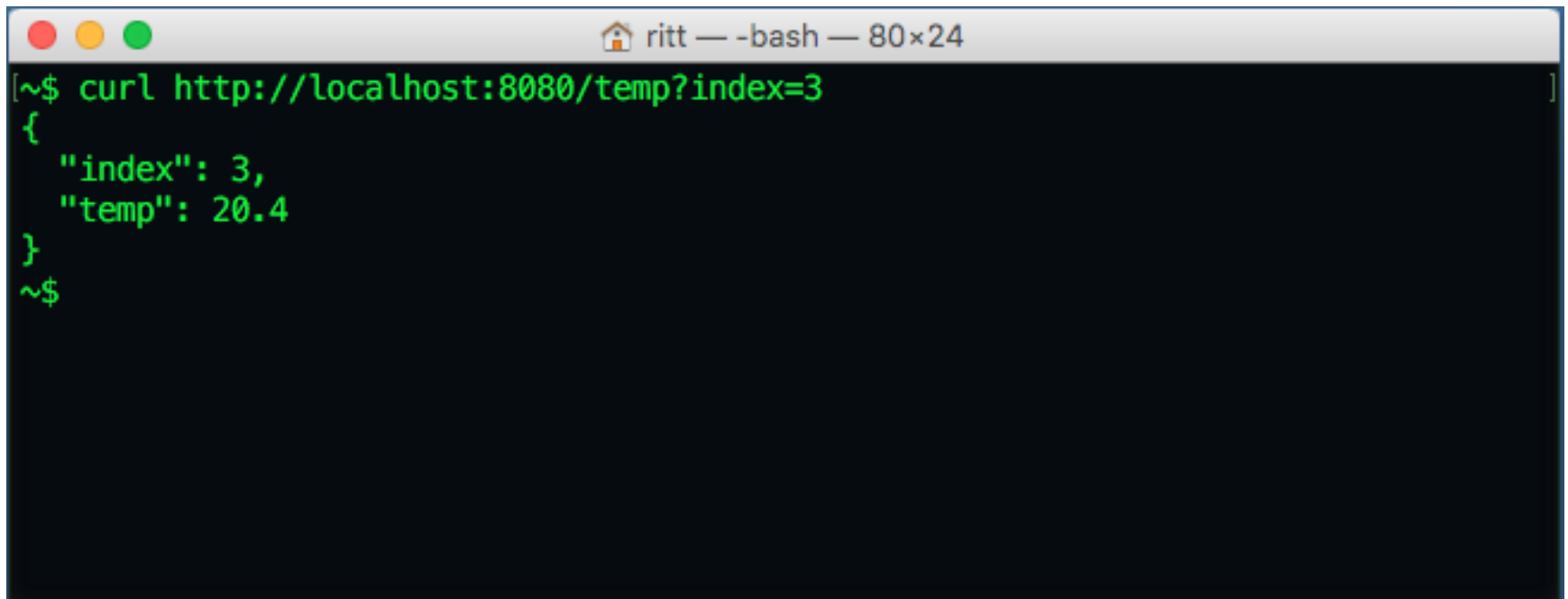
Name	× Headers Preview Response Timing
temp?index=1	<p>▼ Response Headers view parsed</p> <p>HTTP/1.1 200 OK Transfer-Encoding: chunked</p> <p>▼ Request Headers view parsed</p> <p>GET /temp?index=1 HTTP/1.1 Host: localhost:8080 Connection: keep-alive Pragma: no-cache Cache-Control: no-cache Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36 DNT: 1 Accept-Encoding: gzip, deflate, sdch Accept-Language: en-US,en;q=0.8,de-CH;q=0.6</p>

1 / 13 requests | 107 B / 107 B trans..

× Headers Preview Response Timing
<pre>1 { 2 "index": 1, 3 "temp": 27.5 4 } 5</pre>

CURL

- Command line URL program to call servers from the terminal



```
ritt — -bash — 80x24
[~$ curl http://localhost:8080/temp?index=3
]
{
  "index": 3,
  "temp": 20.4
}
~$
```

JSON-RPC

- JSON-RPC implements Remote Procedure Calls:

```
→ {  
  "method": "getLight",  
  "params":  
    {"index": 1},  
  "id": 1234  
}
```

```
← {  
  "result": "on",  
  "error": null,  
  "id": 1234  
}
```

- ID: value of any type to match the response with the request

JSON-RPC Client

```
function rpc_call()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      o = JSON.parse(xhttp.responseText);
      document.getElementById("result").innerHTML = "Result: "+o.result;
    }
  };
  var request = {};
  request.jsonrpc = "2.0";
  request.method = "sum";
  request.params = [];
  request.params[0] = parseFloat(document.getElementById("n1").value);
  request.params[1] = parseFloat(document.getElementById("n2").value);
  request.id = 1;

  xhttp.open("POST", "/json-rpc", true);
  xhttp.setRequestHeader("Content-type", "application/json");
  xhttp.send(JSON.stringify(request));
}
...
<body>
  <input type="text" id="n1">
  <input type="text" id="n2">
  <div id="result">Result:</div>
  <input type="button" value="Add" onClick="rpc_call()">
</body>
```

```
{
  "jsonrpc": "2.0",
  "method": "sum",
  "params": [1, 2],
  "id": 1
}
```

JSON-RPC Sever 1

```
static void ev_handler(struct mg_connection *nc, int ev, void *ev_data)
{
    struct http_message *hm = (struct http_message *) ev_data;
    static const char *methods[] = { "sum", NULL };
    static mg_rpc_handler_t handlers[] = { rpc_sum, NULL };
    char buf[100];

    if (ev == MG_EV_HTTP_REQUEST) {
        if (mg_vcmp(&hm->uri, "/json-rpc") == 0) {
            mg_rpc_dispatch(hm->body.p, hm->body.len, buf, sizeof(buf),
                methods, handlers);
            mg_printf(nc, "HTTP/1.0 200 OK\r\nContent-Length: %d\r\n"
                "Content-Type: application/json\r\n\r\n%s",
                (int) strlen(buf), buf);
            nc->flags |= MG_F_SEND_AND_CLOSE;
        } else {
            // serve static content
            mg_serve_http(nc, hm, s_http_server_opts);
        }
    }
}
```

```
{
  "jsonrpc": "2.0",
  "method": "sum",
  "params": [1, 2],
  "id": 1
}
```

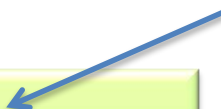
JSON-RPC Server1

```
static int rpc_sum(char *buf, int len, struct mg_rpc_request *req) {
    double sum = 0;
    int i;


    if (req->params[0].type != JSON_TYPE_ARRAY)
        return mg_rpc_create_std_error(buf, len, req,
            JSON_RPC_INVALID_PARAMS_ERROR);

    for (i = 0; i < req->params[0].num_desc; i++) {
        if (req->params[i + 1].type != JSON_TYPE_NUMBER)
            return mg_rpc_create_std_error(buf, len, req,
                JSON_RPC_INVALID_PARAMS_ERROR);

        sum += strtod(req->params[i + 1].ptr, NULL);
    }
    return mg_rpc_create_reply(buf, len, req, "f", sum);
}
```



```
{
  "result":3,
  "id":1
}
```



```
{
  "jsonrpc":"2.0",
  "method":"sum",
  "params":[1,2],
  "id":1
}
```

Raspberry Pi

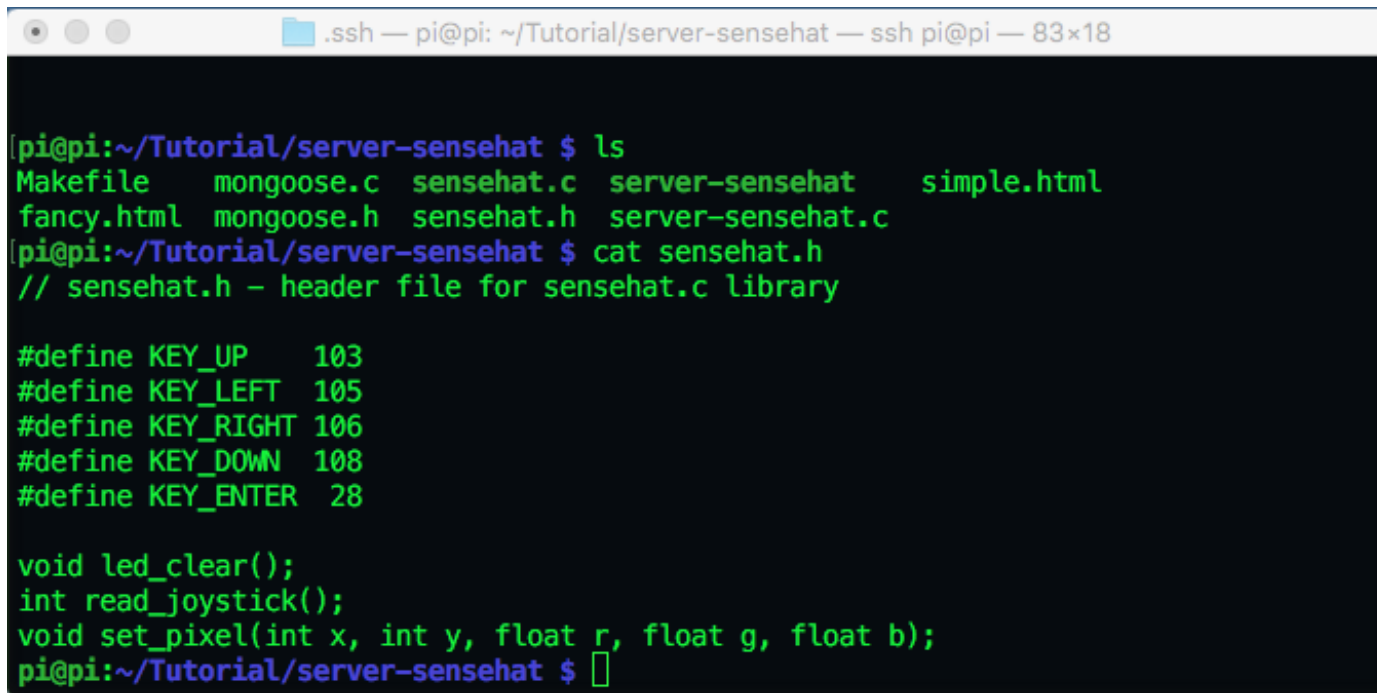
- 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core
- Price: ~35 EUR

Raspberry Pi

- Huge support community
www.raspberrypi.org
- Excellent learning tool
- Many different operating systems, will use Raspian (port of Debian)
- Need housing (8 EUR), Power supply (6 EUR), SD card (11 EUR)
- Many hardware extensions available

Mongoose on Raspberry Pi

- mongoose compiles on Raspberry Pi out of the box
- For hardware access, one needs to link specific libraries to the server



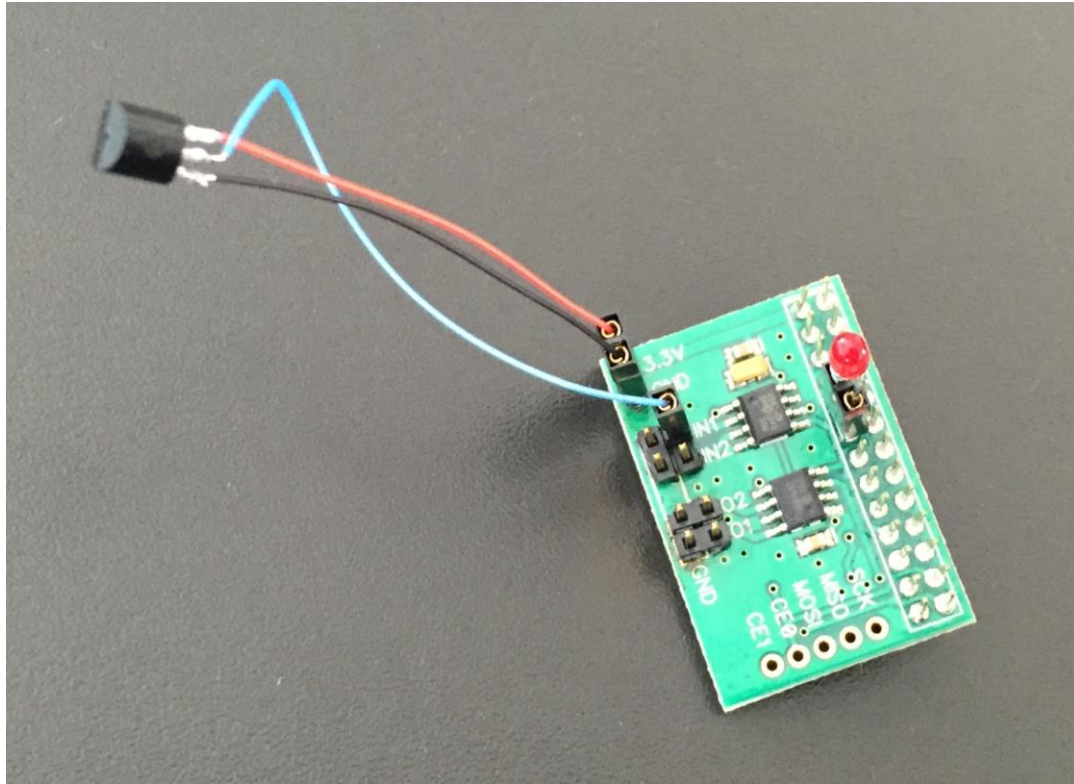
```
.ssh — pi@pi: ~/Tutorial/server-sensehat — ssh pi@pi — 83x18
[pi@pi:~/Tutorial/server-sensehat $ ls
Makefile  mongoose.c  sensehat.c  server-sensehat  simple.html
fancy.html  mongoose.h  sensehat.h  server-sensehat.c
[pi@pi:~/Tutorial/server-sensehat $ cat sensehat.h
// sensehat.h - header file for sensehat.c library

#define KEY_UP    103
#define KEY_LEFT  105
#define KEY_RIGHT 106
#define KEY_DOWN  108
#define KEY_ENTER 28

void led_clear();
int read_joystick();
void set_pixel(int x, int y, float r, float g, float b);
[pi@pi:~/Tutorial/server-sensehat $ ]
```

ADC / DAC Board

- Raspberry Pi “hat” with 2 channel 12 bit ADC and 2 channel 12 bit DAC



LM35 Temperature Sensor



July 1999

LM35 Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of $\pm 1/4^\circ\text{C}$ at room temperature and $\pm 3/4^\circ\text{C}$ over a full -55 to $+150^\circ\text{C}$ temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only $60\ \mu\text{A}$ from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^\circ\text{C}$ temperature range, while the LM35C is rated for a -40° to $+110^\circ\text{C}$ range (-10° with improved accuracy). The LM35 series is available packaged in

hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full -55° to $+150^\circ\text{C}$ range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than $60\ \mu\text{A}$ current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only $\pm 1/4^\circ\text{C}$ typical
- Low impedance output, $0.1\ \Omega$ for 1 mA load

Typical Applications

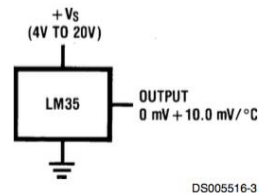
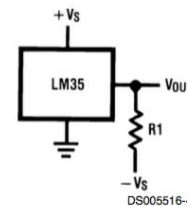


FIGURE 1. Basic Centigrade Temperature Sensor
(+2°C to +150°C)



Choose $R_1 = -V_S/50\ \mu\text{A}$
 $V_{\text{OUT}} = +1,500\ \text{mV}$ at $+150^\circ\text{C}$
 $= +250\ \text{mV}$ at $+25^\circ\text{C}$
 $= -550\ \text{mV}$ at -55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Server 1

```
static int rpc_readtemp(char *buf, int len, struct mg_rpc_request *req)
{
    unsigned char spi_buf[3];
    double t;

    spi_buf[0] = 0x01; // Start
    spi_buf[1] = 0xA0; // SGL/DIFF=1, ODD/SING=0, MS/BF=1
    spi_buf[2] = 0x00;

    wiringPiSPIDataRW(spi_fd0, spi_buf, 3);

    // convert to Volts
    t = (((spi_buf[1] & 0x0F) << 8) | spi_buf[2])/4096.0 * 3.3;

    // convert to deg C
    t *= 100;

    // round to one digit
    t = (int)(t*10+0.5)/10.0;

    return mg_rpc_create_reply(buf, len, req, "1")
}
```

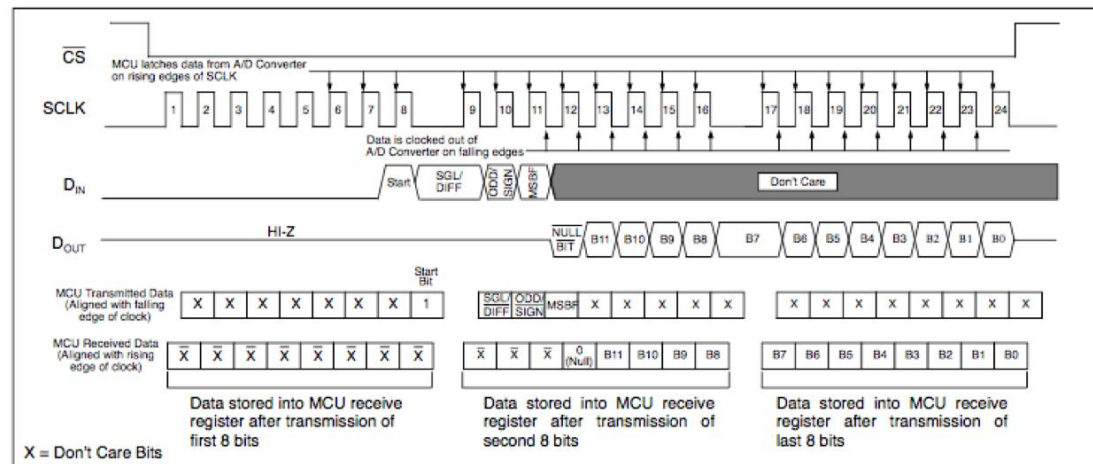


FIGURE 6-1: SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).

Server 2

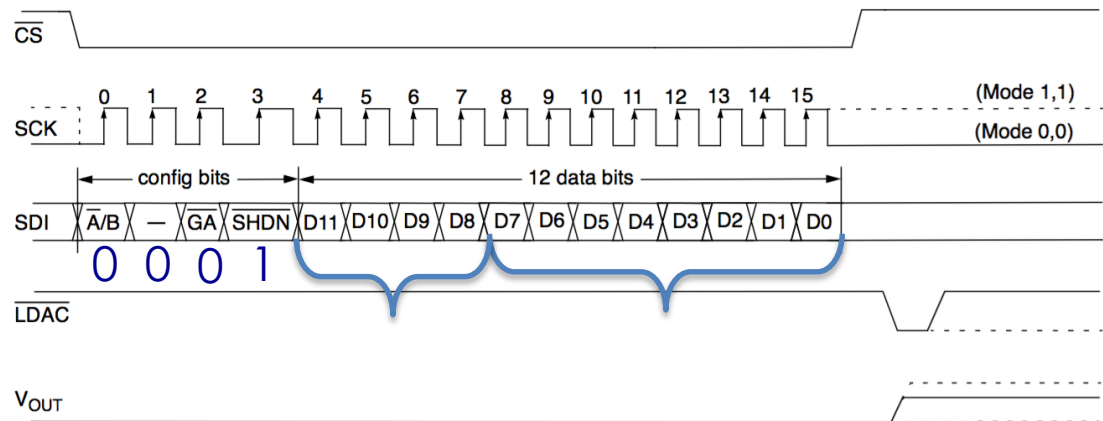
```
static int rpc_sethv(char *buf, int len, struct mg_rpc_request *req)
{
    double hv;
    unsigned char spi_buf[2];
    unsigned int d;

    hv = atof(req->params[1].ptr);

    d = (unsigned int) (hv / 2 / 2.048 * 4096);
    spi_buf[0] = 0x10 | (d >> 8); // MSB, Gain=2, /SHDN=1, Channel A
    spi_buf[1] = d & 0xFF;      // LSB
    wiringPiSPIDataRW(spi_fd1, spi_buf, 2);

    // toggle /LDAC
    digitalWrite(6, LOW);
    digitalWrite(6, HIGH);

    return mg_rpc_create_reply(buf, len)
}
```



Server 3

```
static void ev_handler(struct mg_connection *nc, int ev, void *ev_data)
{
    struct http_message *hm = (struct http_message *) ev_data;
    static const char *methods[] = { "sethv", "readtemp", NULL };
    static mg_rpc_handler_t handlers[] = { rpc_sethv, rpc_readtemp, NULL };
    char buf[1000];

    if (ev == MG_EV_HTTP_REQUEST) {
        if (mg_vcmp(&hm->uri, "/json-rpc") == 0) {
            mg_rpc_dispatch(hm->body.p, hm->body.len, buf, sizeof(buf),
                methods, handlers);
            mg_printf(nc, "HTTP/1.0 200 OK\r\nContent-Length: %d\r\n"
                "Content-Type: application/json\r\n\r\n%s",
                (int) strlen(buf), buf);
            nc->flags |= MG_F_SEND_AND_CLOSE;
        } else {
            // serve static content
            mg_serve_http(nc, hm, s_http_server_opts);
        }
    }
}
```

Client Temperature

```
<!DOCTYPE html>
<html>
<head>
  <title>Temperature Display</title>

<style>
  body {
    margin: 0;
  }
  #temp {
    height: 300px;
    background: yellow;
    background: linear-gradient(red,
yellow);

    font-family: sans-serif;
    font-size: 250px;
    text-align: center;
  }
</style>
```

```
<script>

window.setTimeout(load, 1000);

function load()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      o = JSON.parse(xhttp.responseText);
      document.getElementById("temp").innerHTML = o.result;
      window.setTimeout(load, 1000);
    }
  };

  var request = {};
  request.jsonrpc = "2.0";
  request.method = "readtemp";
  request.id = 1;

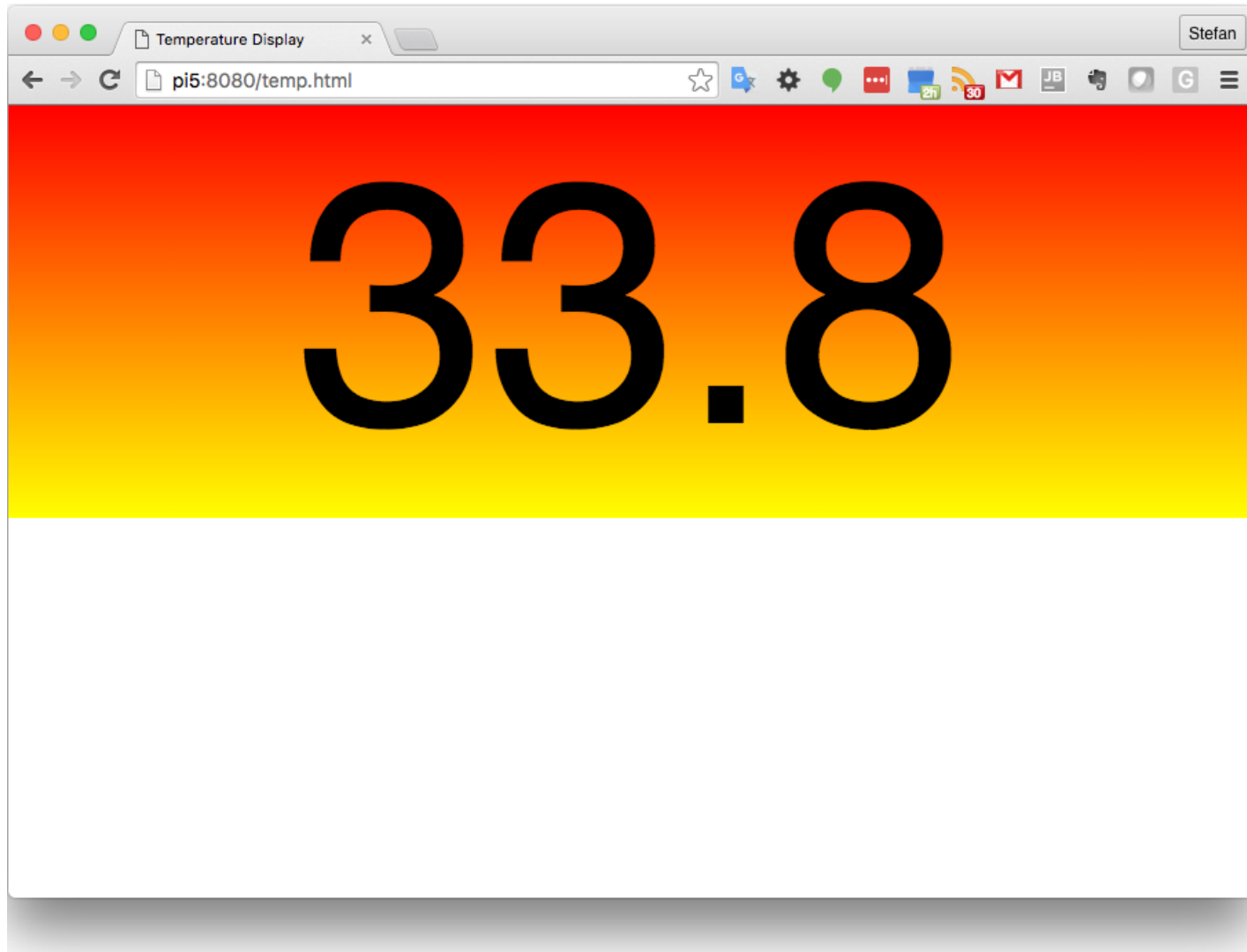
  xhttp.open("POST", "/json-rpc", true);
  xhttp.setRequestHeader("Content-type", "application/json");
  xhttp.send(JSON.stringify(request));
}

</script>
</head>

<body onload="load();">
  <div id="temp">0.0</div>
</body>

</html>
```


Client Temperature



Client HV

```
<!DOCTYPE html>
<html>
<head>
  <title>HV</title>

<style>
  body {
    font-family: sans-serif;
    font-size: 24px;
    margin: 30px;
  }
  input {
    line-height: 24px;
    font-size: 24px;
    -webkit-appearance:none;
  }
</style>
<script>
```

```
function rpc_sethv()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      o = JSON.parse(xhttp.responseText);
    }
  };

  var request = {};
  request.jsonrpc = "2.0";
  request.method = "sethv";
  request.params = [];
  request.params[0] = document.getElementById("hv").value;
  request.id = 1;

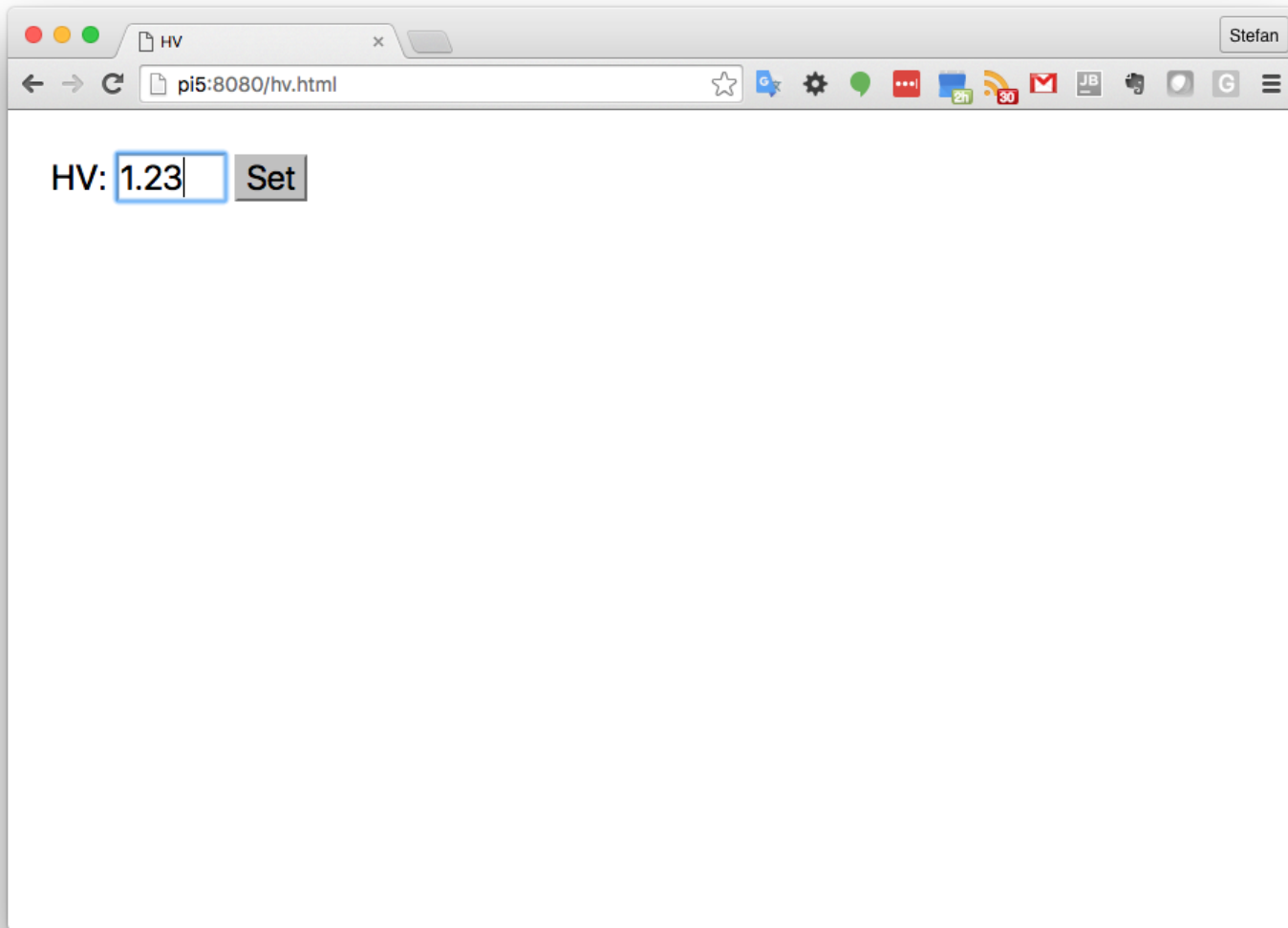
  xhttp.open("POST", "/json-rpc", true);
  xhttp.setRequestHeader("Content-type", "application/json");
  xhttp.send(JSON.stringify(request));
}

</script>
</head>

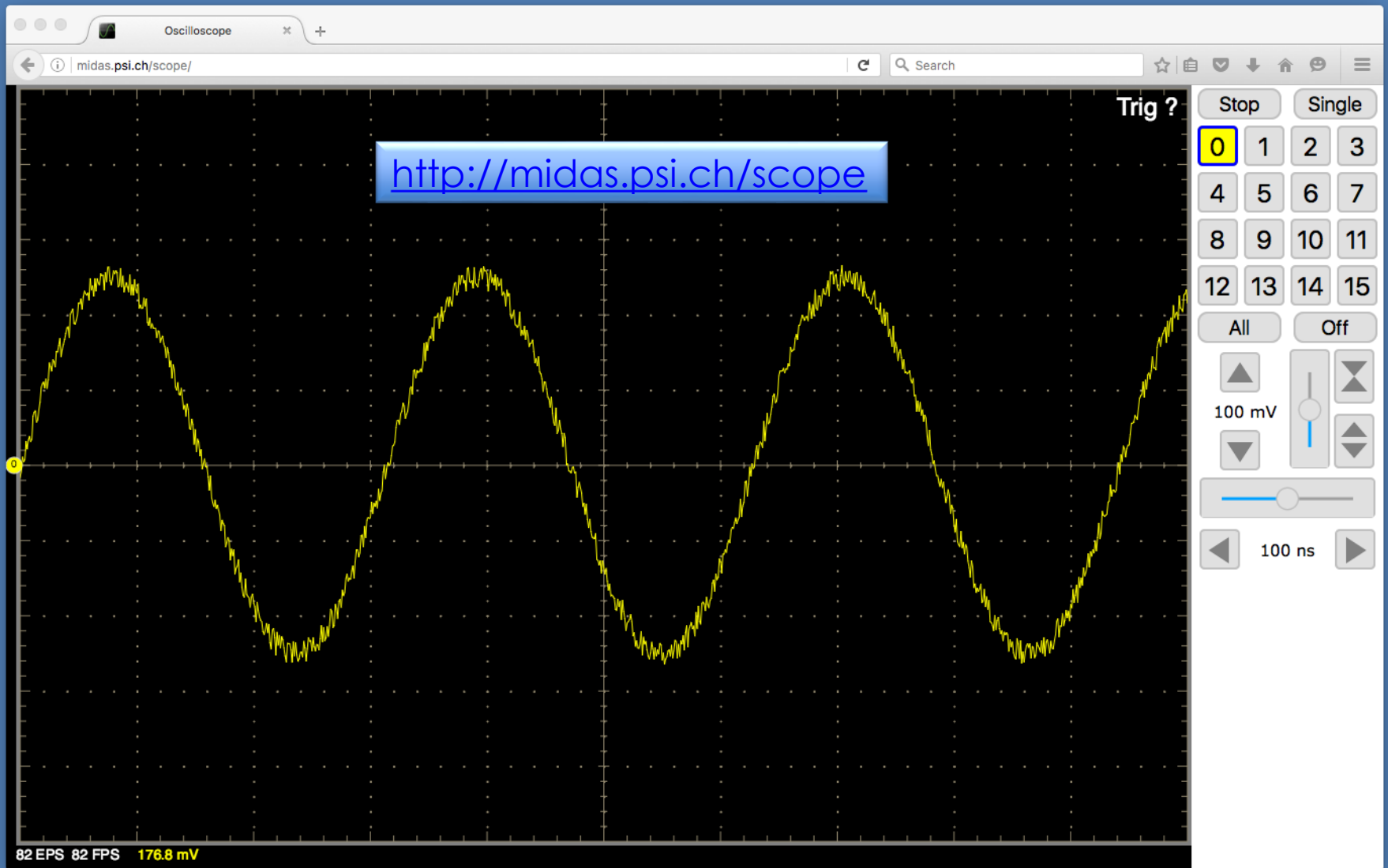
<body>
  HV: <input type="text" id="hv" value="0" size="5">
  <input type="button" value="Set" onclick="rpc_sethv()">
</body>

</html>
```

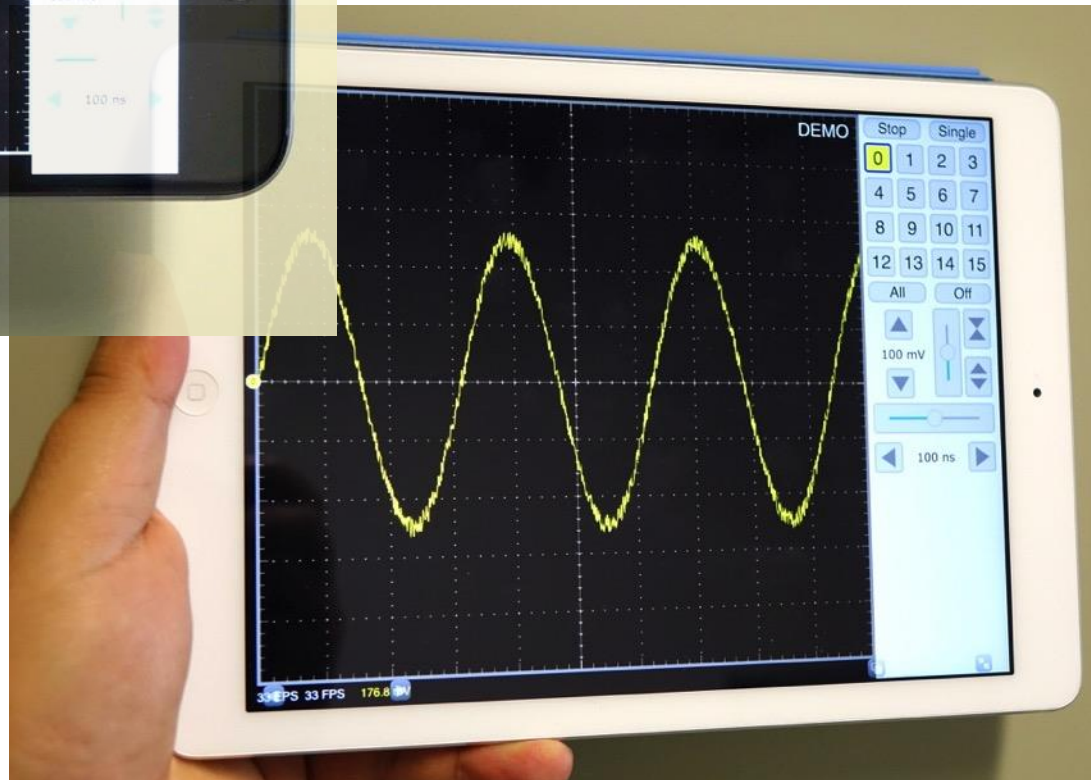
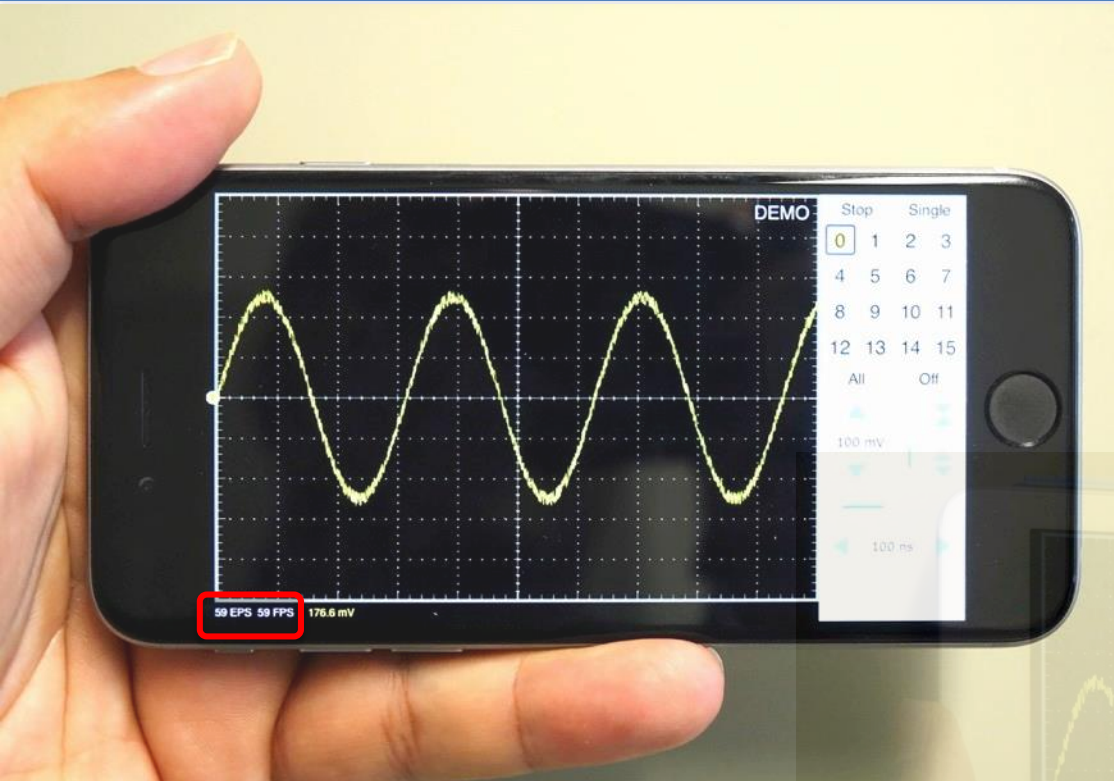
Client HV



Oscilloscope Application



Smartphone and Tablet



Conclusion: Future of visualization

- Experiment data can be displayed on almost any browser
 - HTML5 is standardized through W3C
 - No software to install, no libraries required
 - Central updates of software
 - Only raw (binary) data needs to be transferred (as opposed to remote desktop applications)
 - Perfect for remote monitoring and control
 - Easy to learn (Scope took me one week from scratch)
- Web server can be incorporated into any web device (“Internet of Things (IoT)”)
- Slides, programs, ... on Dropbox:
<https://db.tt/WWkBEhMQ>