



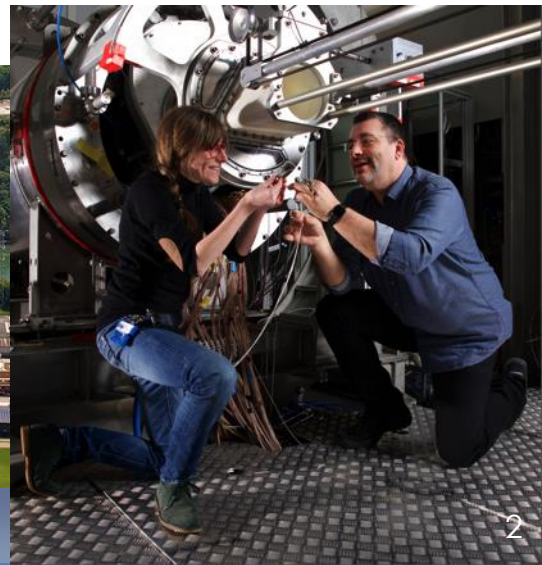
Stefan Ritt, Paul Scherrer Institute, Switzerland

REAL-TIME DATA VISUALIZATION AND CONTROL USING MODERN WEB TECHNOLOGIES

3RD REAL TIME SCHOOL, CAPE TOWN, SOUTH AFRICA, 2018

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 (elog.psi.ch/elog)
MIDAS DAQ system (midas.triumf.ca)
DRS chip (www.psi.ch/drs)

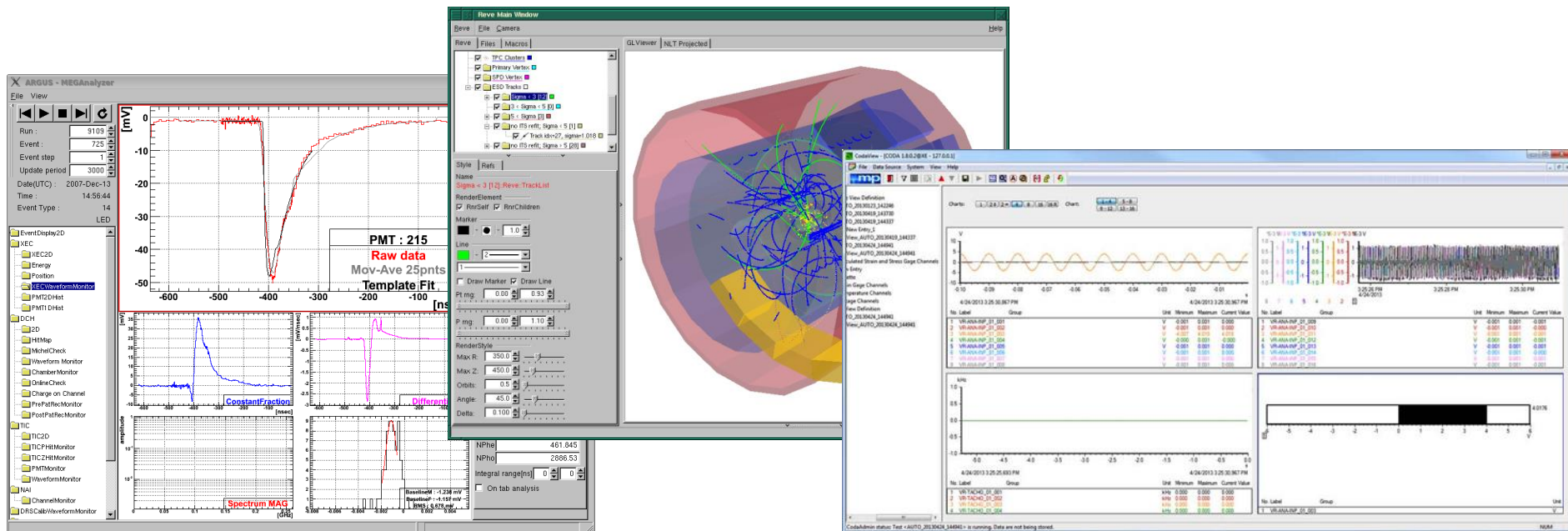


About this Lecture

- 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
- Please ask questions any time

Experimental Data Visualization and Control

- Some Visualization & Control require Real-Time
- 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

The screenshot displays the WaveDAQ Oscilloscope web interface. The main display area shows 16 channels of waveforms, each with a corresponding colored channel number (0-15) on the left. A central 'About' dialog box is open, displaying the following information:

WaveDAQ Oscilloscope
Built May 20 2016
Stefan Ritt
Paul Scherrer Institute

The interface includes a control panel on the right with the following settings:

- Device: wd020
- Buttons: Stop, Single
- Channel selection: 0-15 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
- Buttons: All, Off
- Vertical scale: 1 V
- Horizontal scale: 5 ns
- Buttons: Config, About
- Trigger: 0 mV, Trigger type: normal (selected), auto
- Analog Front-end: Gain: 1 (selected), 10, 100, PZC; Mode: DRS (selected), ADC; Input range: -0.5 V ... +0.5 V
- Checkboxes: Enable calibration clock, Connect inputs to calib. source, Power calib. source
- DC: 0 mV
- Sampling Speed: 5 GSPS, Actual: 4.995 GSPS
- Voltage Calibration: Apply Cell Calibration, Apply Readout Calibration, Apply Gain Calibration, Correct Range Offset, Remove Spikes
- Buttons: Execute Voltage Calibration
- Buttons: Execute Time Calibration

A blue banner at the bottom of the interface contains the URL: <https://elog.psi.ch/scope>

At the bottom of the waveform display, the following data is shown: 15 EPS 16 FPS T=45.3 C 359.71 mV 359.46 mV 358.54 mV 359.09 mV 355.70 mV 359.01 mV 359.74 mV 360

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

HTML (content) + CSS (style)

Hypertext Markup Language

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

Cascading Style Sheets

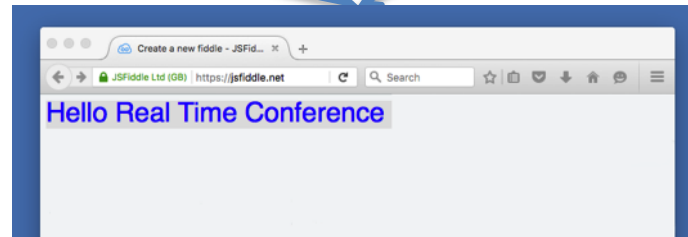
Selector
Declaration
Declaration
Declaration
Declaration

```
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>
```

First HTML Page

HTML:

```
<p>Welcome to the <span>Real Time</span> School.</p>
```

```
<p>Do you enjoy the  
<span class="subtle">Real Time</span> School?</p>
```

```
<p>Will you come to the next  
<span class="subtle" id="next">Real Time</span> School?</p>
```

CSS:

```
body {  
    font-family: sans-serif;  
    font-size:120%;  
    color: blue;  
}
```

```
span { color: red; }
```

```
.subtle {  
    font-style: italic; }
```

```
#next { color: black; }
```

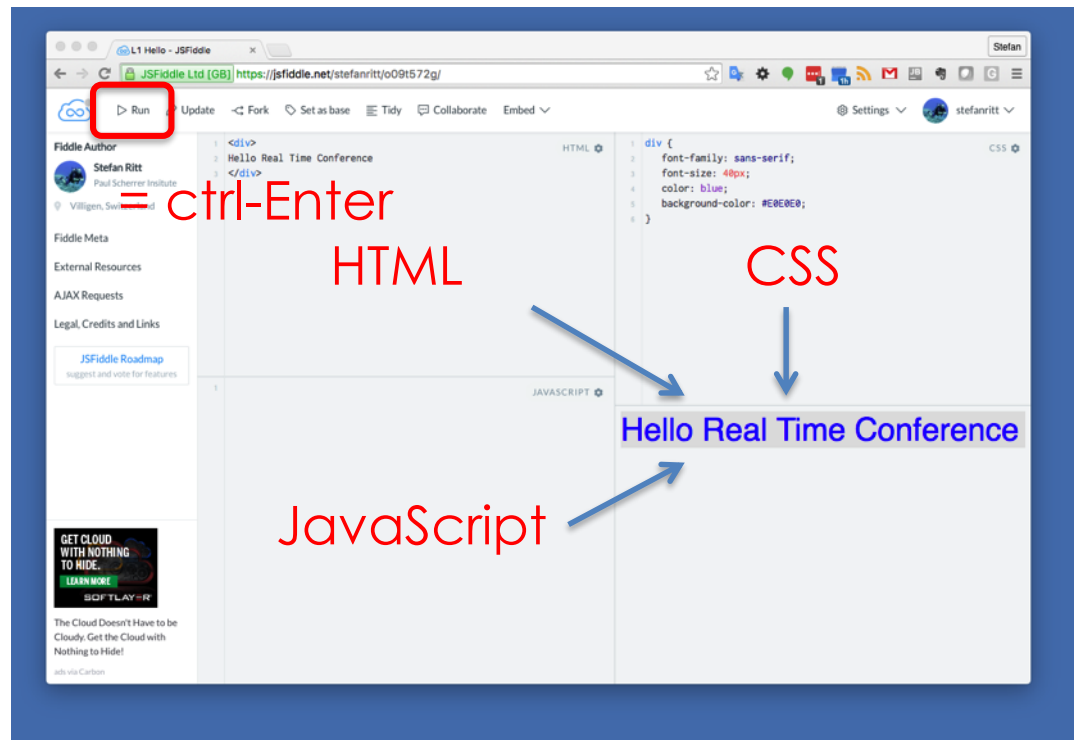
Welcome to the **Real Time** School.

Do you enjoy the **Real Time** School?

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

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 browser window with the URL `jsfiddle.net/stefanritt/o09t572g/`. The page content is defined by the following code:

```
HTML
1 <p>
2   Welcome to the <span>Real Time</span> School.
3 </p>
4
5 <p>
6   Do you enjoy the <span class="subtle">Real
7     Time</span> School?
8 </p>
9
10 <p>
11   Will you come to the next <span class="subtle"
12     id="next">Real Time</span> School?
13 </p>
14
CSS
1 body {
2   font-family: sans-serif;
3   font-size: 200%;
4   color: blue;
5   background-color: #E0E0E0;
6 }
7
8 span {
9   color: red;
10 }
11
12 .subtle {
13   font-style: italic;
14 }
15
16 #next {
17   color: black;
18 }
```

The rendered output in the preview area is:

Welcome to the **Real Time** School.

Do you enjoy the *Real Time* School?

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

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 for the selected element:

```
Filter Styles
element {
  #next {
    color: black;
  }
  .subtle {
    font-style: italic;
  }
  span {
    color: red;
  }
  Inherited from body
  body {
    font-family: sans-serif;
    font-size: 180%;
    color: blue;
  }
}
```

Red arrows point to the `color: black`, `font-style: italic`, `color: red`, and `font-size: 180%` properties in the CSS Rules pane.

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;  
}
```

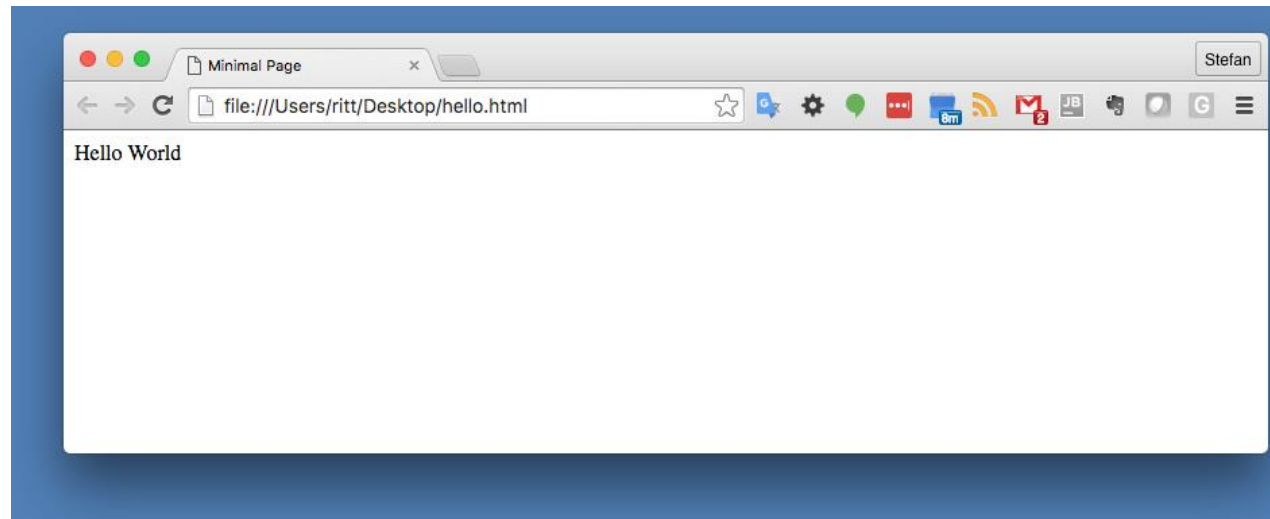
http://www.w3schools.com/cssref/css_selectors.asp

HTML - CSS

- Do separate content and style!
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

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 is divided into three main sections: a left sidebar with project details, a central code editor, and a right preview pane.

Left Sidebar:

- Fiddle Author
- Fiddle Meta
- L3 Form
- Simple input form with text fields and check boxes

HTML Code (lines 1-30):

```
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 Code (lines 1-4):

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

Preview Pane:

The rendered form includes:

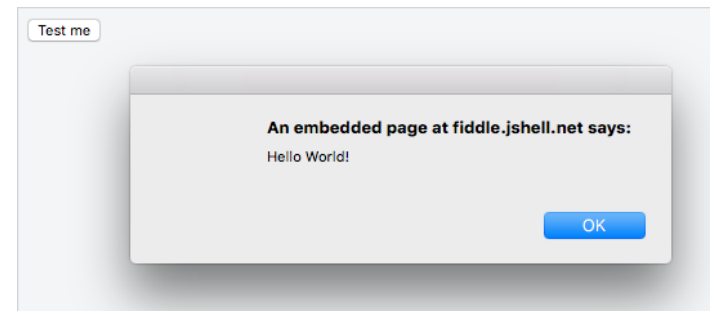
- Hometown:
- Gender: Male Female
- Age:
- Enter comment here
- Submit

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

HTML ▼

```
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>
```

JavaScript + No-Library (pure JS) ▼

```
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:
8   " + sum;
9 }
```

CSS ▼

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

X:

Y:

Result: 3

Debugging JavaScript

The screenshot shows a web browser window with a JSFiddle editor. The editor contains HTML, CSS, and JavaScript code. The JavaScript code defines a function `add()` that takes two inputs, `x` and `y`, and returns their sum. The browser's developer tools are open, showing the `Sources` panel with the `add()` function paused at line 49. The `Console` panel shows the output of the function, and the `Scope` panel shows the local variables `sum`, `x`, and `y`.

```
<p>
  X:
  <input type="text" id="x">
</p>
<p>
  Y:
  <input type="text" id="y">
</p>
function add() {
  var x = document.getElementById("x").value;
  var y = document.getElementById("y").value;
  var sum = parseInt(x) + parseInt(y);
  //alert(sum);
  document.getElementById("result").innerHTML = "Result: " + sum;
}
```

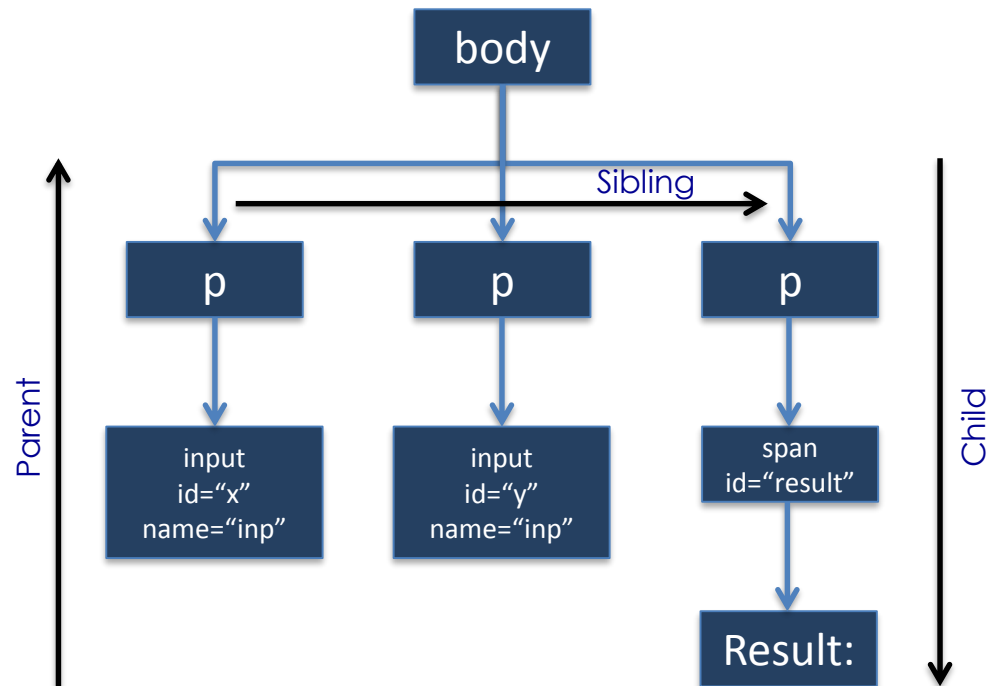
```
function add() {
  47
  48   var x = document.getElementById("x").value;
  49   var y = document.getElementById("y").value;
  50   var sum = parseInt(x) + parseInt(y);
  51   //alert(sum);
  52
  53   document.getElementById("result").innerHTML = "Result: " + sum;
  54 }
  55
  56 //]]>
  57 </script>
  58
  59 </head>
  60
  61
  62
  63
  64 }
```

```
var sum: 3
this: Window
  x: "1"
  y: "2"
```

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

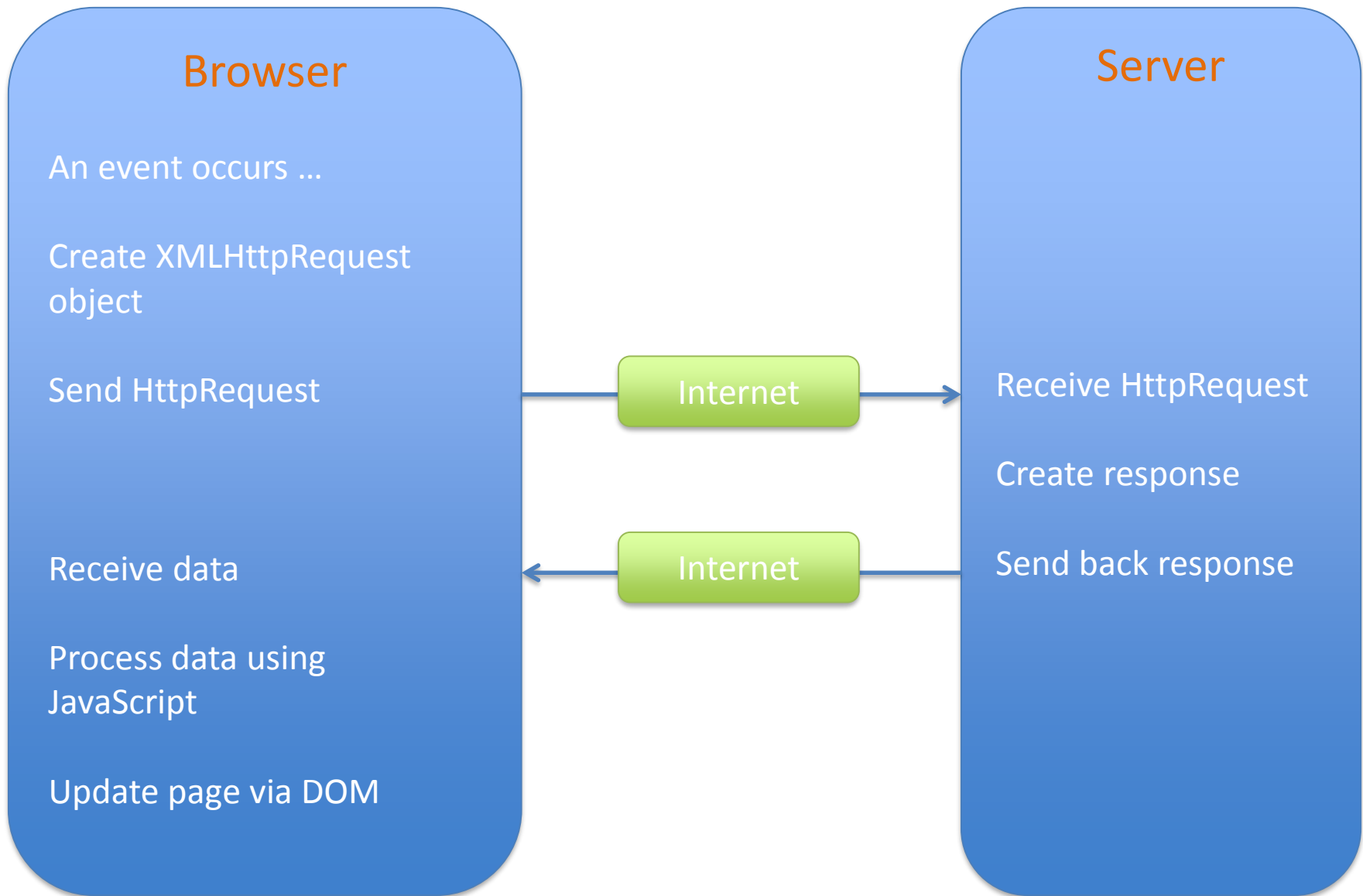
```
$(".subtle").css("color", "red");
```

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

HTML ▼

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

JavaScript + No-Library (pure JS) ▼

```
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 =
6       xhttp.responseText;
7     }
8   };
9   xhttp.open("POST", "/echo/html/", true);
10  xhttp.send("html=Response received&delay=3");
11 }
```

CSS ▼

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

Response received

Send

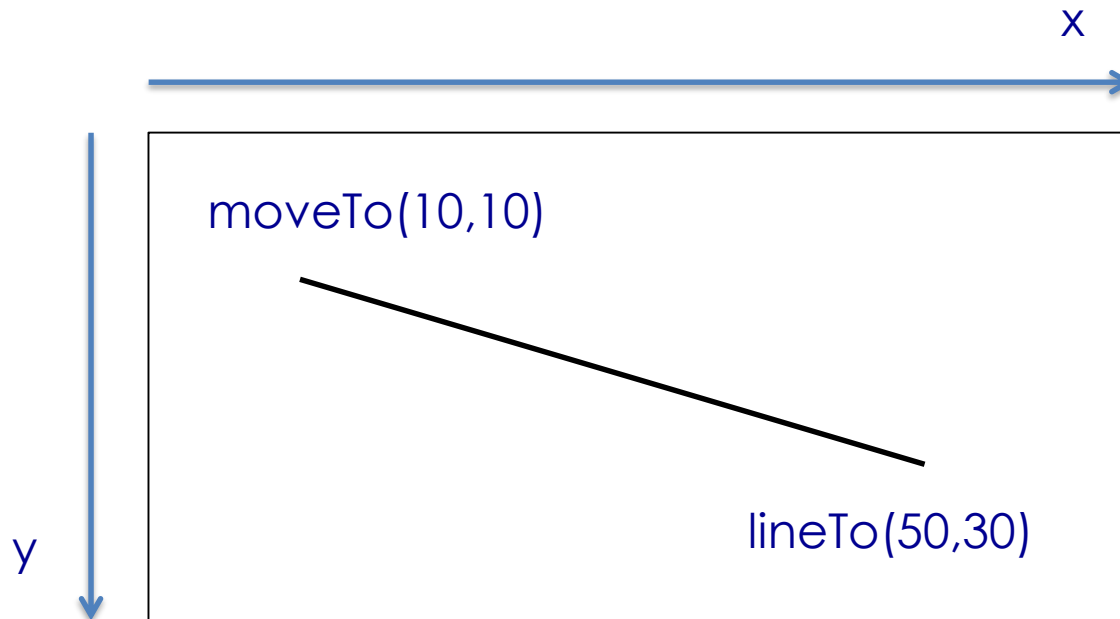
JSON

- 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

HTML ▼

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

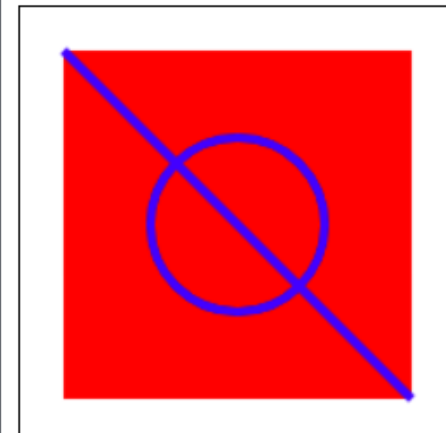
CSS ▼

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

JavaScript + No-Library (pure JS) ▼

```
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
```

Canvas Demo



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>
```

Custom Slider

HTML ▾

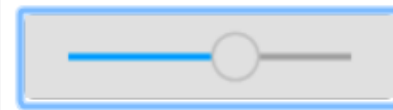
```
1 <button name="slider" class="slider" type="button" data-  
  update="update()"></button>  
2 <div id="value"></div>  
3
```

CSS ▾

```
1 ▾ body {  
2   font-family:sans-serif;  
3 }  
4 ▾ .slider {  
5   margin: 3px;  
6   width: 174px;  
7   height: 40px;  
8   border-radius: 5px;  
9   padding: 0px;  
10 }  
11
```

JavaScript + No-Library (pure JS) ▾

```
1 function Controls() {} // empty constructor  
2  
3 Controls.prototype.init = function()  
4 {  
5   // scan DOM for sliders  
6   this.ctrlSlider = document.getElementsByName("slider");  
7  
8   for (var i=0 ; i<this.ctrlSlider.length ; i++) {  
9     var cvs = document.createElement("canvas");  
10    var sl = this.ctrlSlider[i];  
11    cvs.width = sl.clientWidth;  
12    cvs.height = sl.clientHeight;  
13    sl.appendChild(cvs);  
14    sl.canvas = cvs;  
15  
16    sl.position = 0.5; // slider position 0..1  
17    sl.addEventListener("click",
```



0.59

Dialog Box

HTML ▼

```
1 <div>
2   <p>
3     <input type="button" value="Dialog"
4       onClick="showDialog();"
5     </p>
6   <p id="bg">Background</p>
7 </div>
8 <div id="dialog" class="dialog">
9   <div class="dialogTitle"
10    id="dialogTitle">Dialog Title</div>
11  <div class="dialogContents">
12    Dialog contents
13    <br /> Dialog contents
14  </div>
15 </div>
```

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  font-family: sans-serif;
11  background-color: #F0F0F0;
12  width: 200px;
13  text-align: center;
14  padding: 0;
```

JavaScript + No-Library (pure JS) ▼

```
1 function showDialog() {
2   var e = document.getElementById("dialog");
3   e.style.display = "block";
4   e.style.left =
5     document.documentElement.clientWidth / 2 -
6     e.offsetWidth / 2 + "px";
7   e.style.top =
8     document.documentElement.clientHeight / 2 -
9     e.offsetHeight / 2 + "px";
10
11   window.addEventListener("mousedown",
12     dialogDrag, true);
13   window.addEventListener("mousemove",
14     dialogDrag, true);
```

Dialog



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%

Method of defining a repeating linear or radial color gradient as a CSS image. unprefixed: 77.74%

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 <small>+</small>	
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						

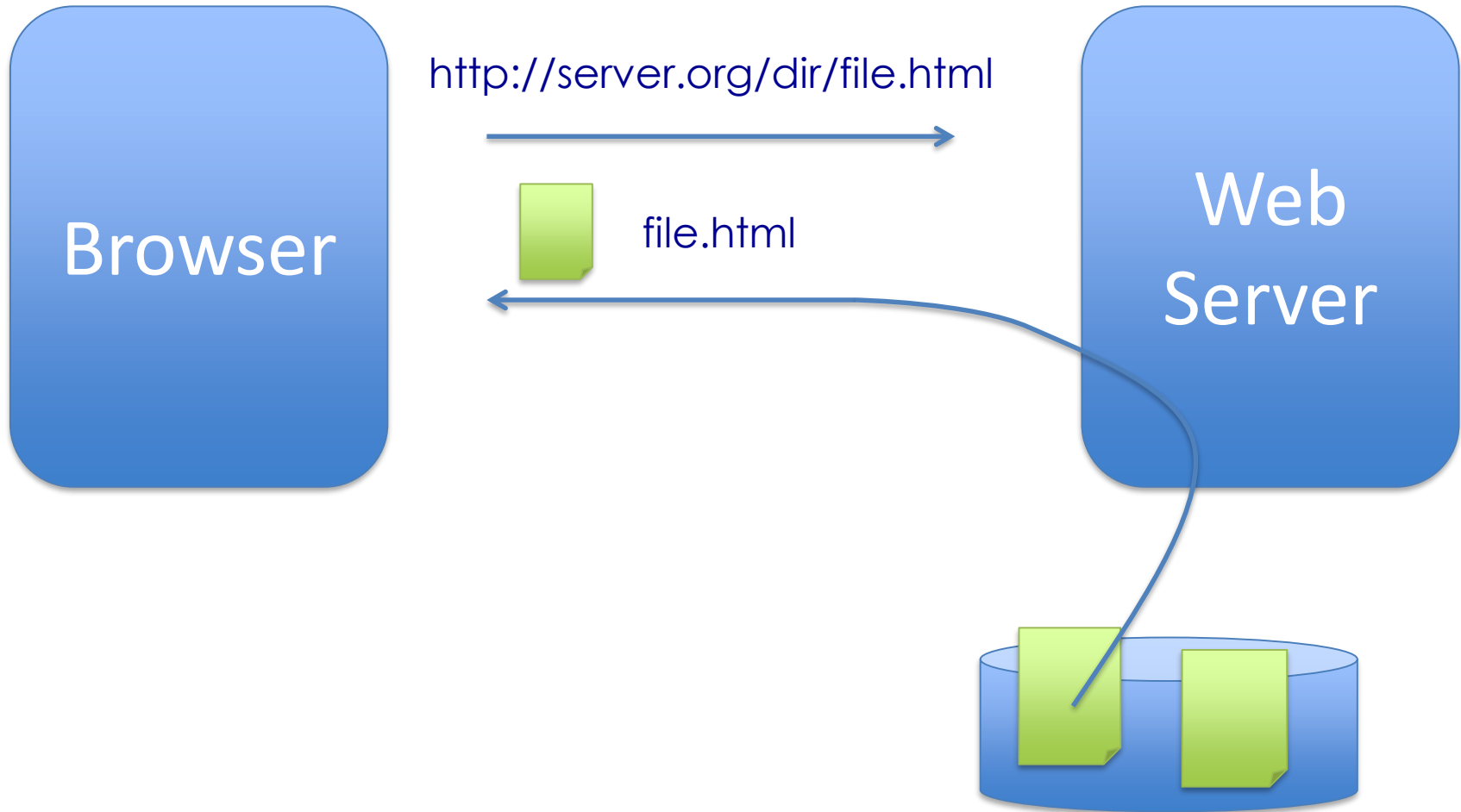
[Notes](#)
[Known issues \(1\)](#)
[Resources \(4\)](#)
[Feedback](#)

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

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

Web servers

www.apache.org

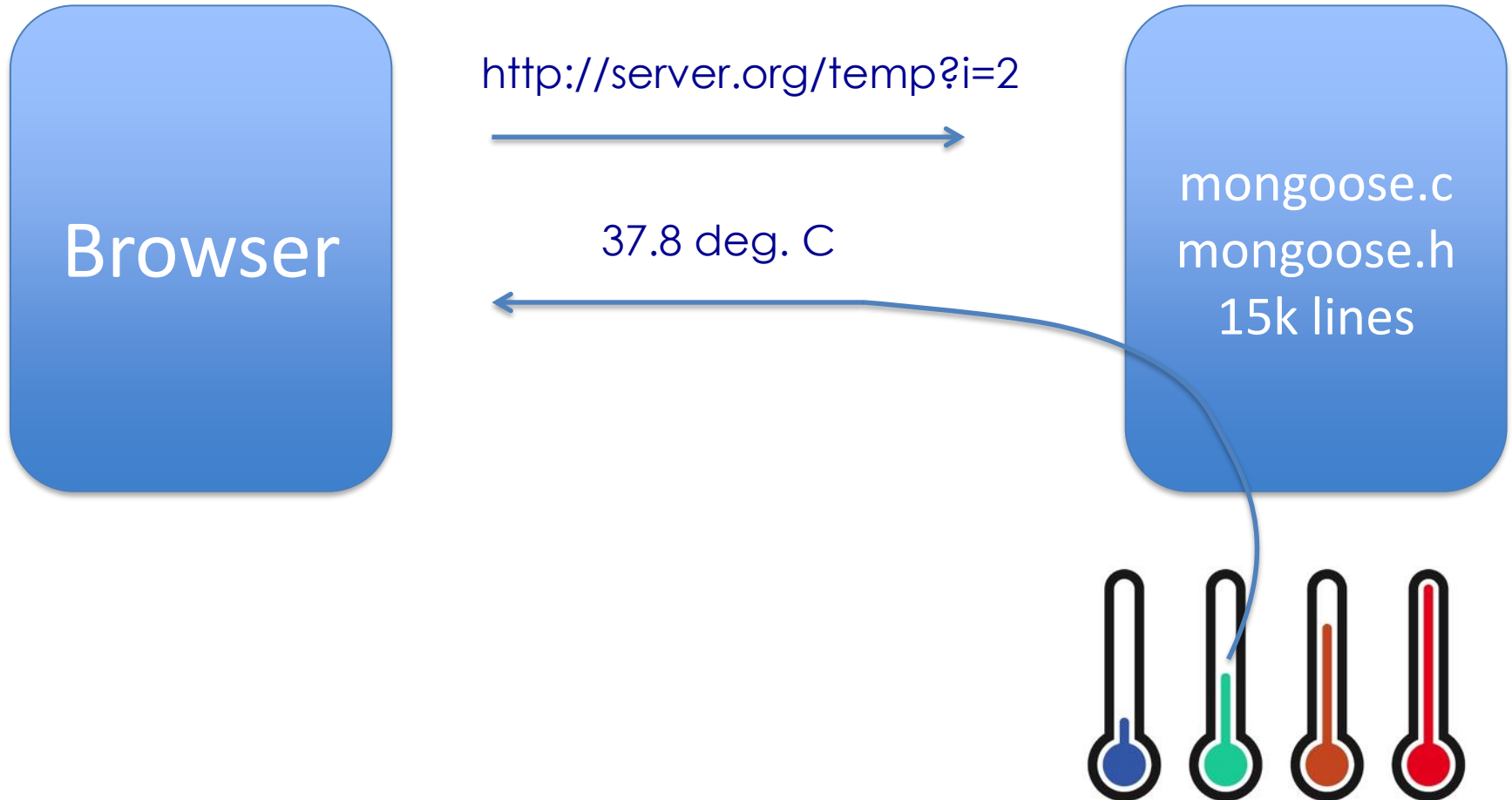


Web Servers

- 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);
    }

    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);

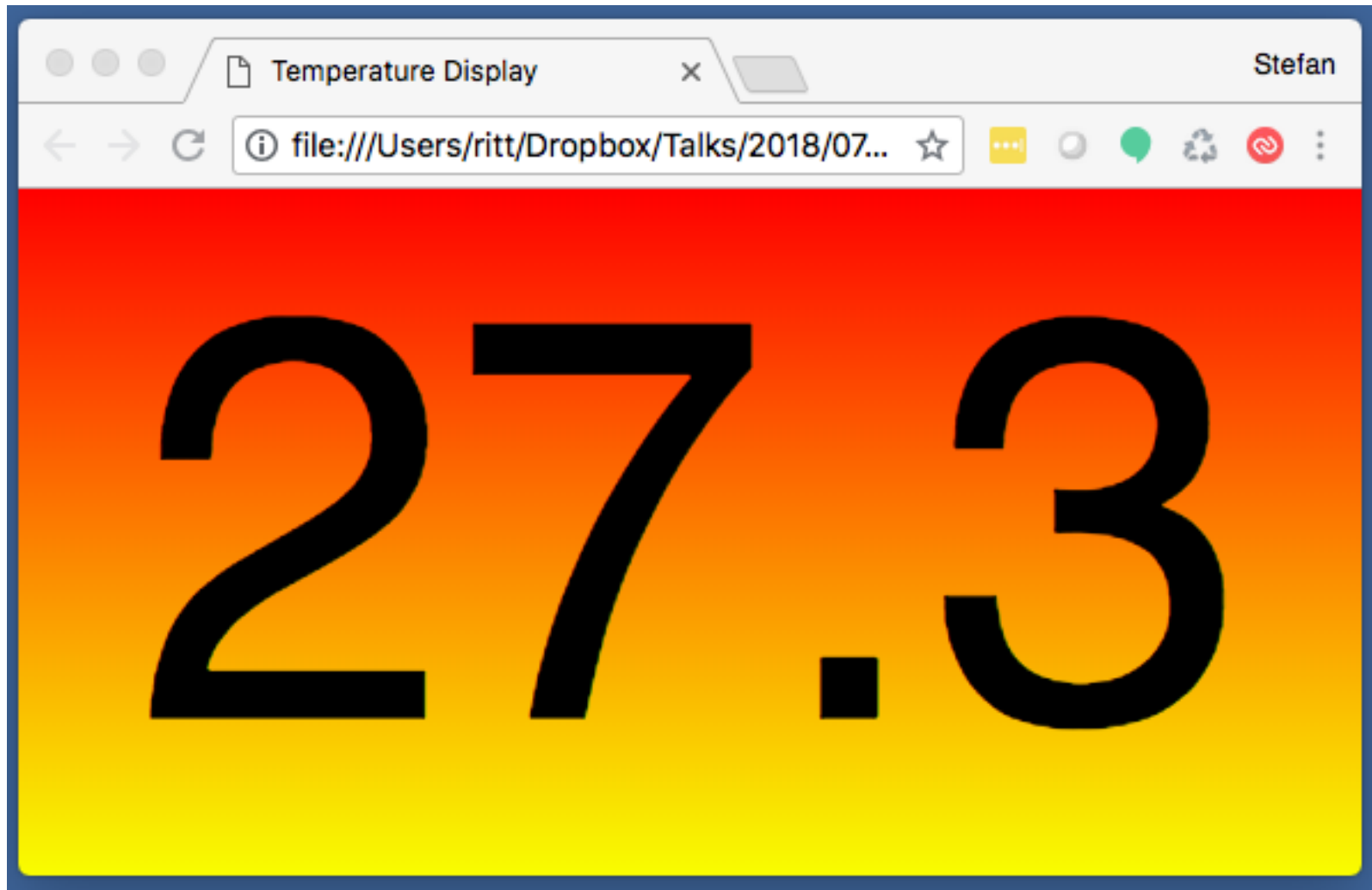
function load()
{
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 &&
xhttp.status == 200) {
      o = JSON.parse(xhttp.responseText);
      document.getElementById("temp").innerHTML
= o.temp;
      window.setTimeout(load, 1000);
    }
  };
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

</script>
</head>

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

</html>
```

Temperature Web Page



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
- Exercises with Raspberry Pi will use port 8080
- 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 response body contains a JSON object: `{ "index": 1, "temp": 20 }`. A blue arrow points to the 'Explanation' section of the waterfall chart.

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

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

1 / 13 requests | 107 B / 107 B transferred | Finish: 286 ms | DOMContentLoaded: 283 ms | Load: 283 ms

HTTP Request and Response

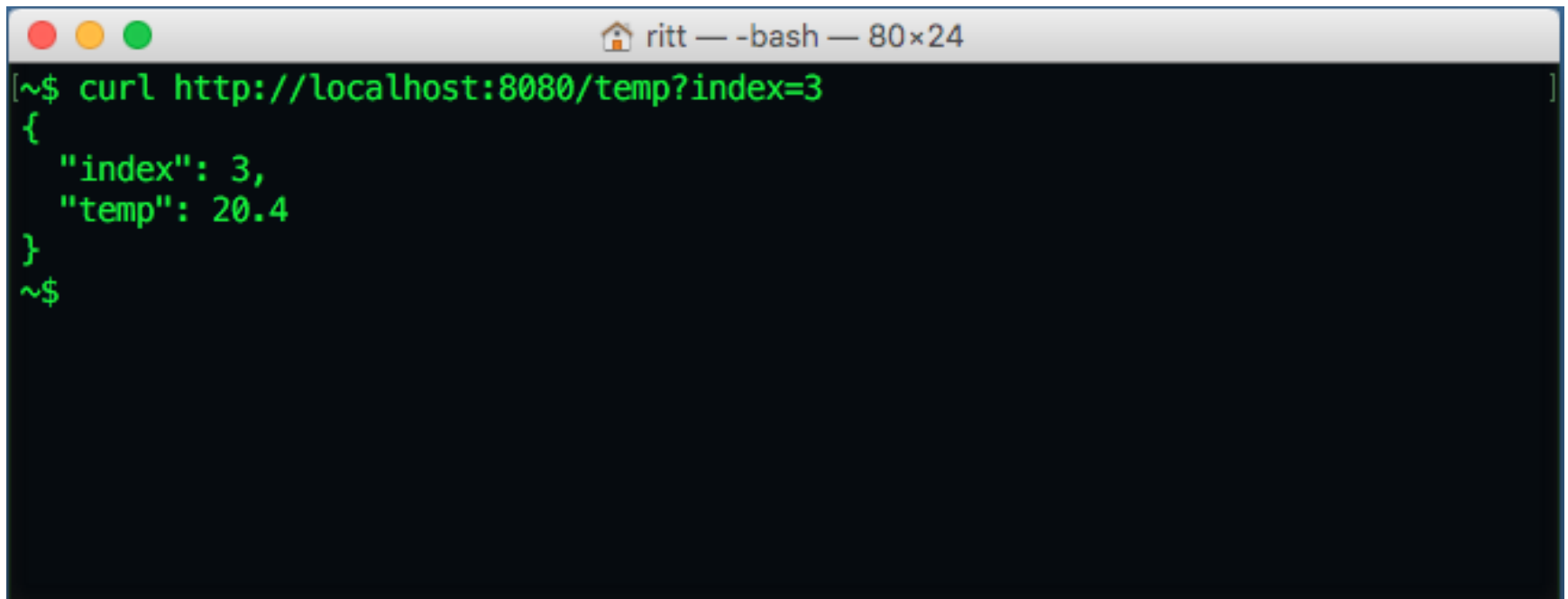
Name	× Headers Preview Response Timing
temp?index=1	<p>▼ Response Headers view parsed</p> <p>HTTP/1.1 200 OK Transfer-Encoding: chunked</p> <p>▼ Request Headers view parsed</p> <p>GET /temp?index=1 HTTP/1.1 Host: localhost:8080 Connection: keep-alive Pragma: no-cache Cache-Control: no-cache Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;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
}
```

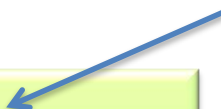
JSON-RPC Server1

```
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);
}
```

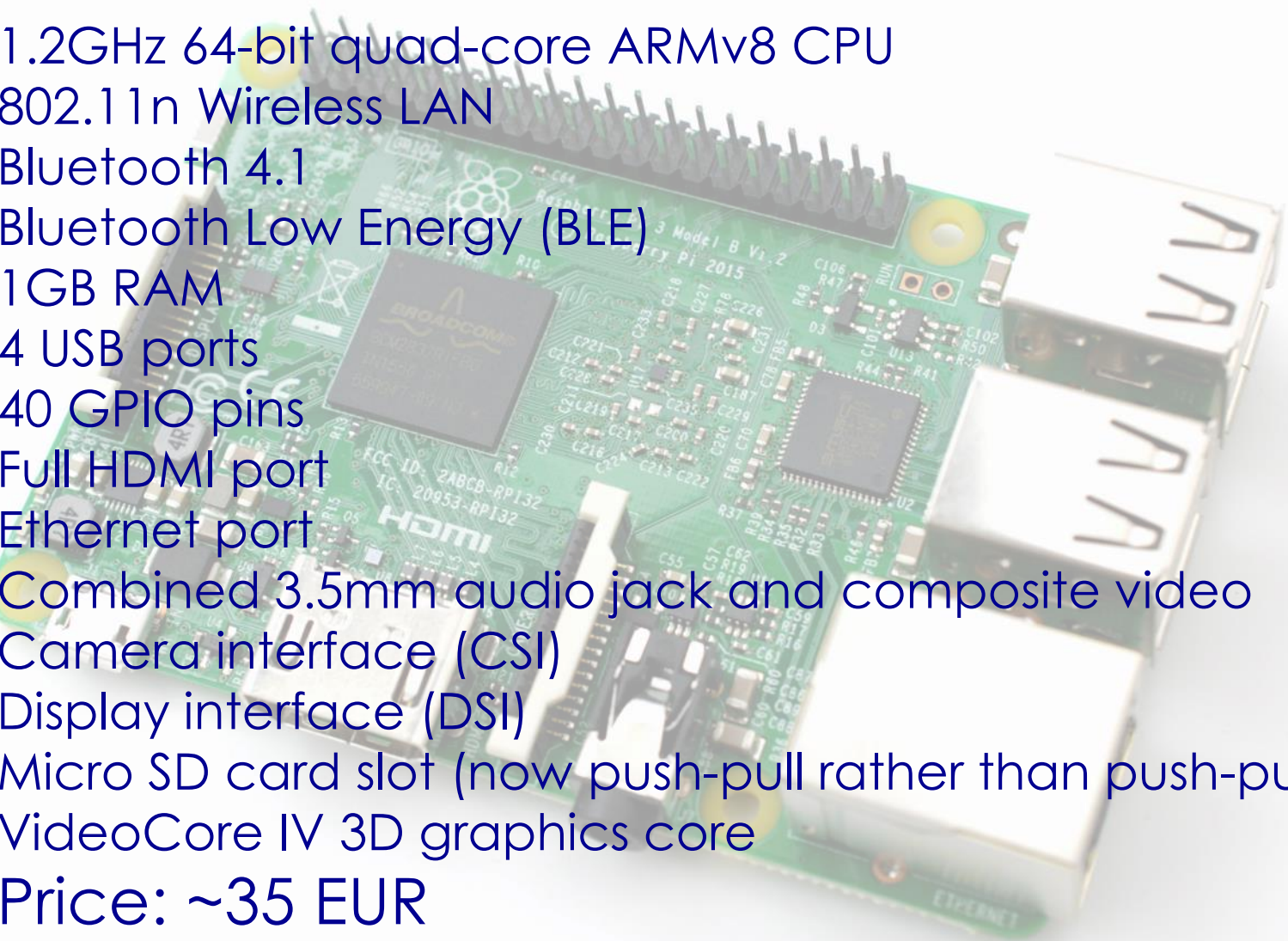


```
{
  "result":3,
  "id":1
}
```



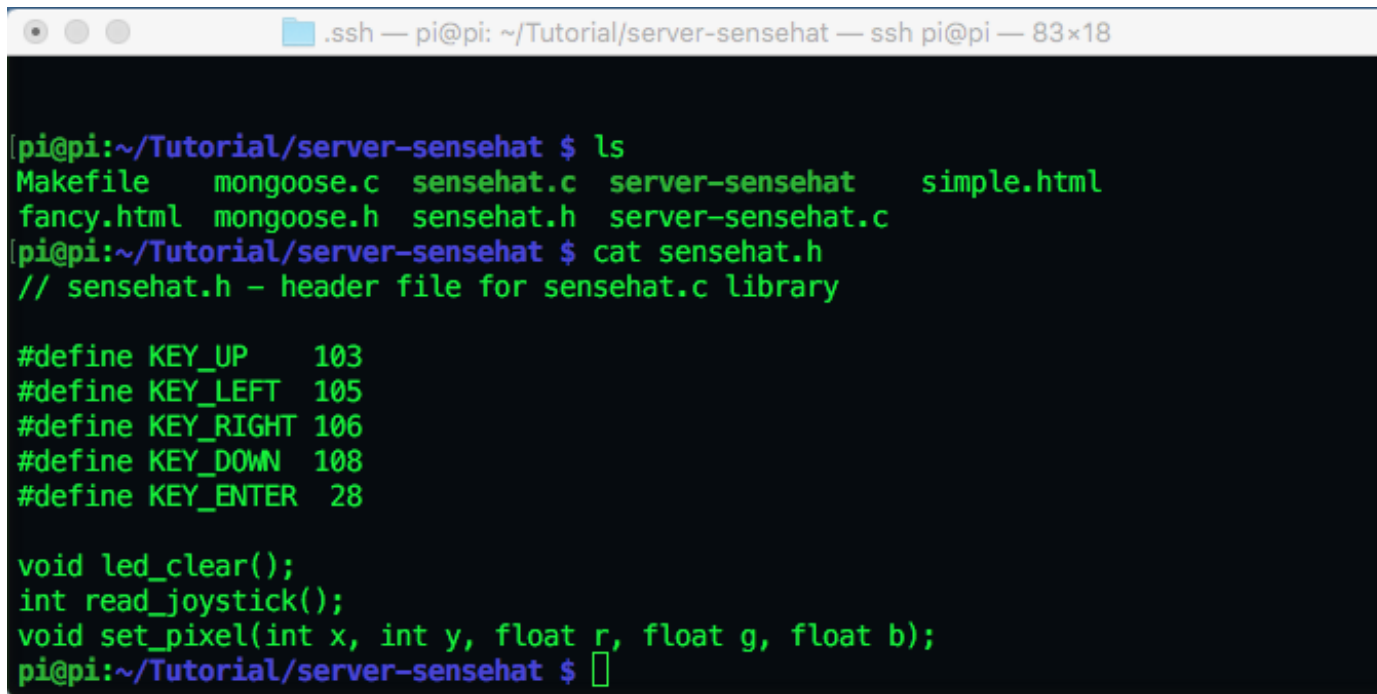
```
{
  "jsonrpc":"2.0",
  "method":"sum",
  "params":[1,2],
  "id":1
}
```

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
- 

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



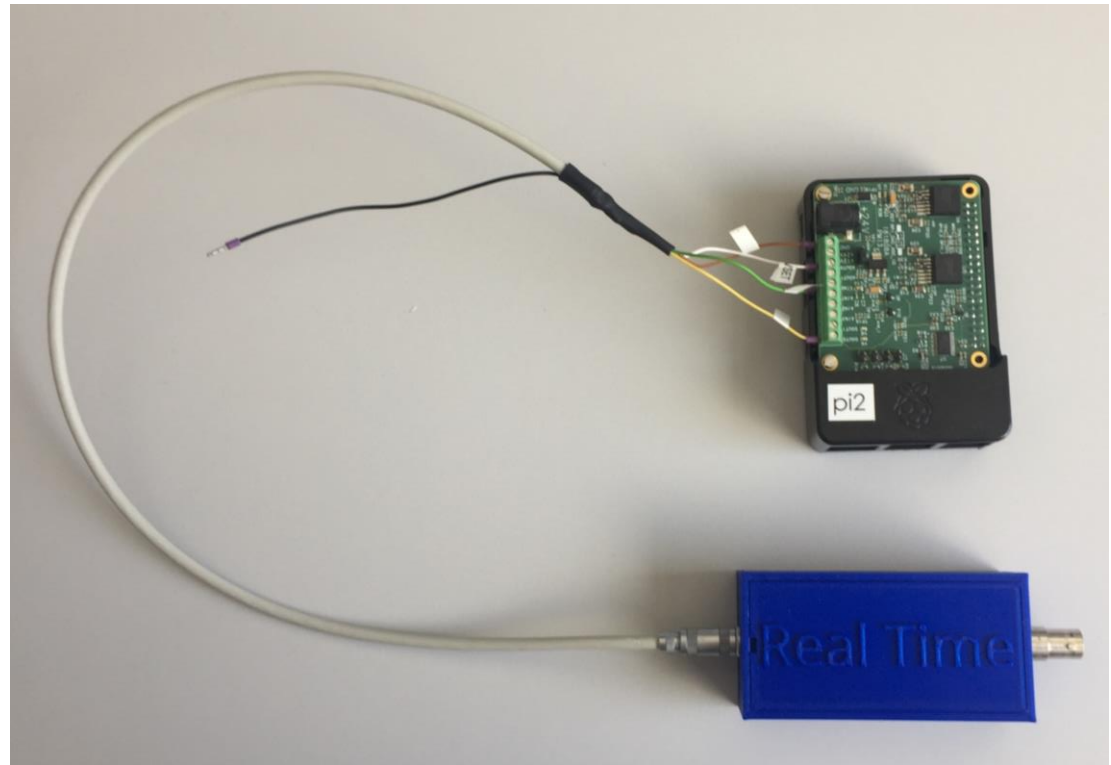
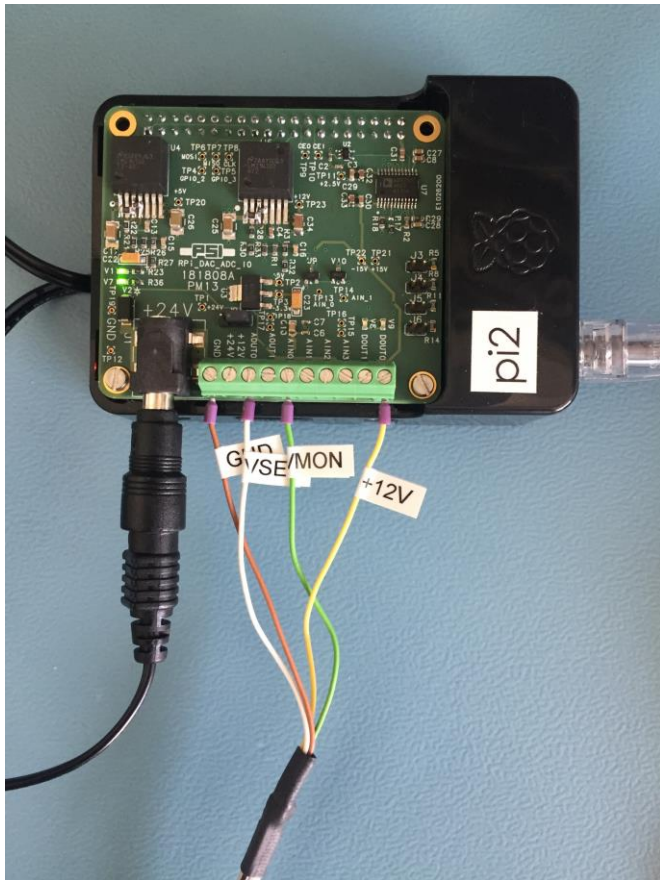
```
[pi@pi:~/Tutorial/server-sensehat $ ls
Makefile  mongoose.c  sensehat.c  server-sensehat  simple.html
fancy.html  mongoose.h  sensehat.h  server-sensehat.c
[pi@pi:~/Tutorial/server-sensehat $ cat sensehat.h
// sensehat.h - header file for sensehat.c library

#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 4 channel 16 bit ADC and 2 channel 16 bit DAC



Programming DAC through SPI

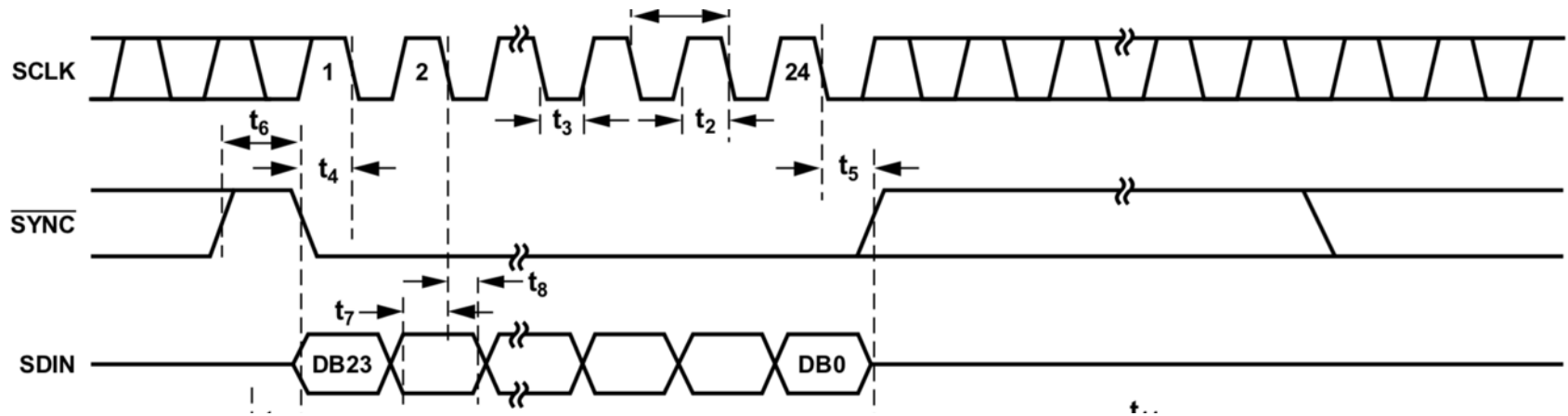


Table 18. Programming the [AD5752](#) DAC Register

	MSB							LSB
R/W	Zero	REG2	REG1	REG0	A2	A1	A0	DB15 to DB0
0	0	0	0	0	DAC address			16-bit DAC data

0x00

0x1234

Server 1

```
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 / 2000 * 65535);
    spi_buf[0] = 0x00;
    spi_buf[1] = d >> 8;    // MSB
    spi_buf[2] = d & 0xFF; // LSB
    wiringPiSPIDataRW(spi_fd0, spi_buf, 3);

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

Server 2

```
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", NULL };
    static mg_rpc_handler_t handlers[] = { rpc_sethv, 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 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;

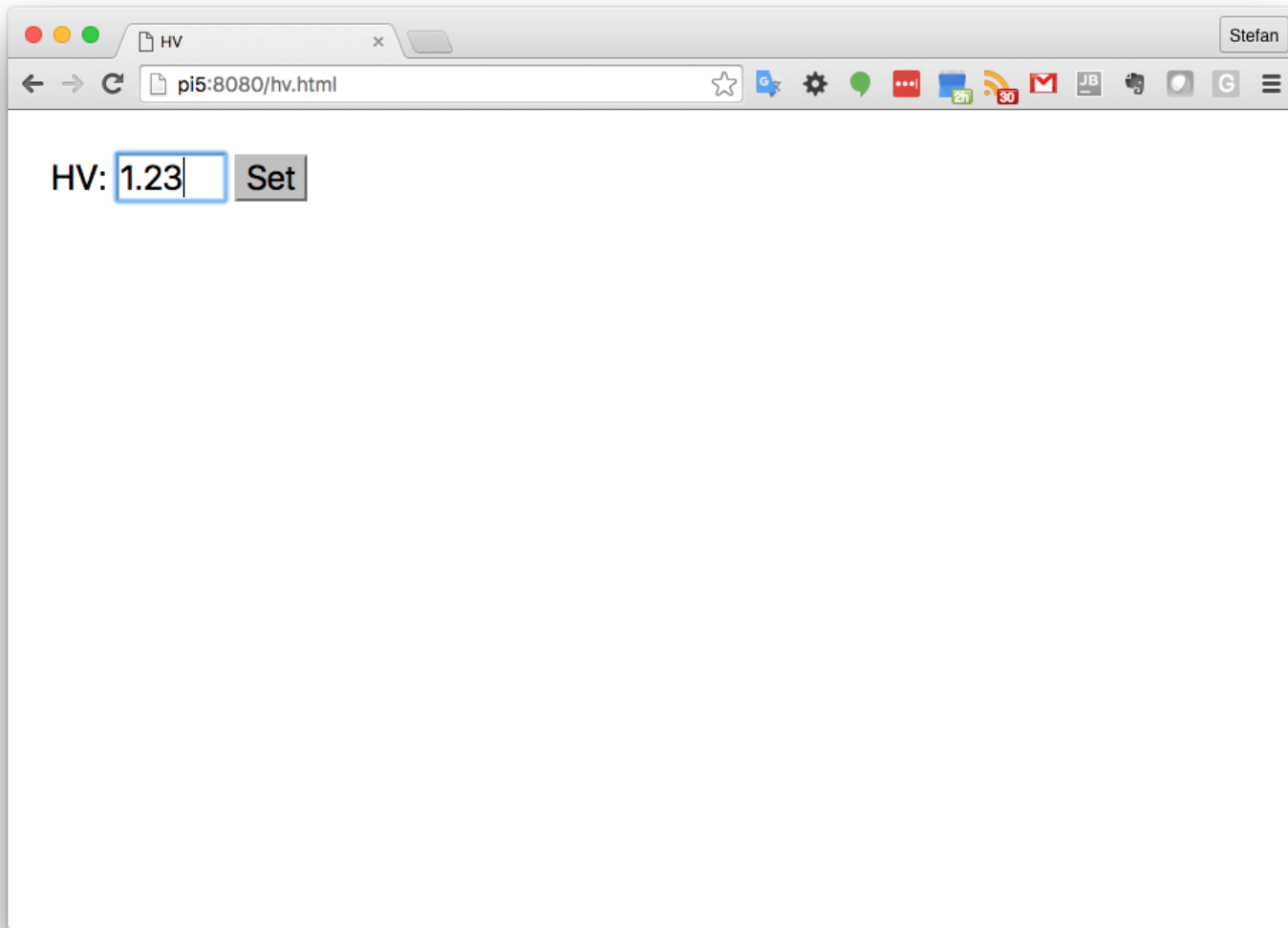
  xhttp.open("POST", "/json-rpc", true);
  xhttp.setRequestHeader("Content-type", "application/json");
  xhttp.send(JSON.stringify(request));
}

</script>
</head>

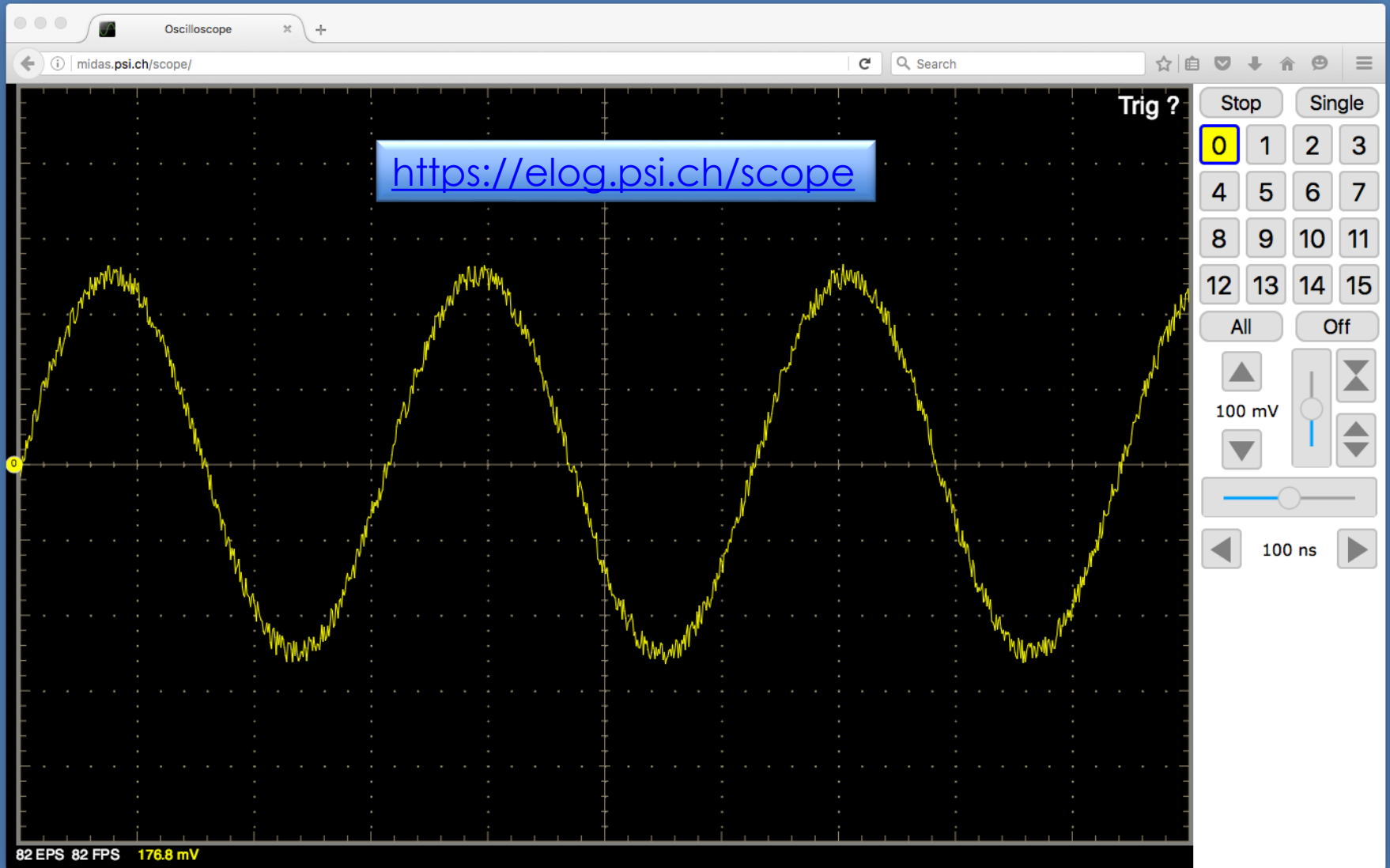
<body>
  HV: <input type="text" id="hv" value="0" size="5">
  <input type="button" value="Set" onclick="rpc_sethv()">
</body>

</html>
```

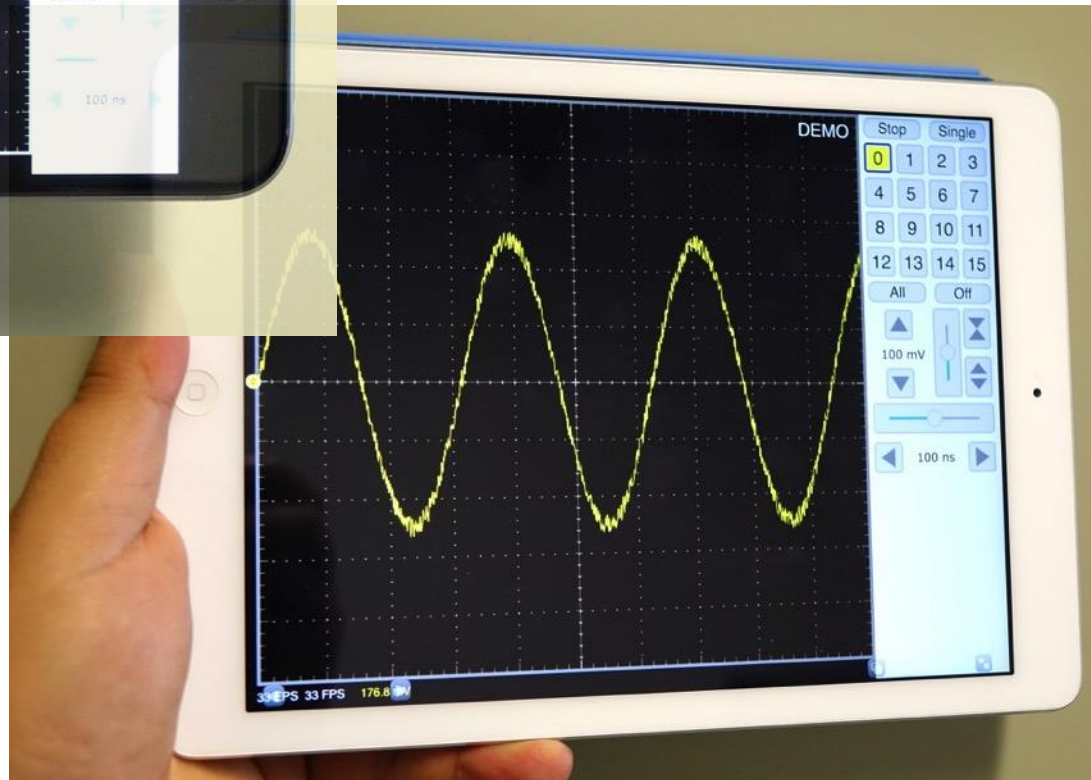
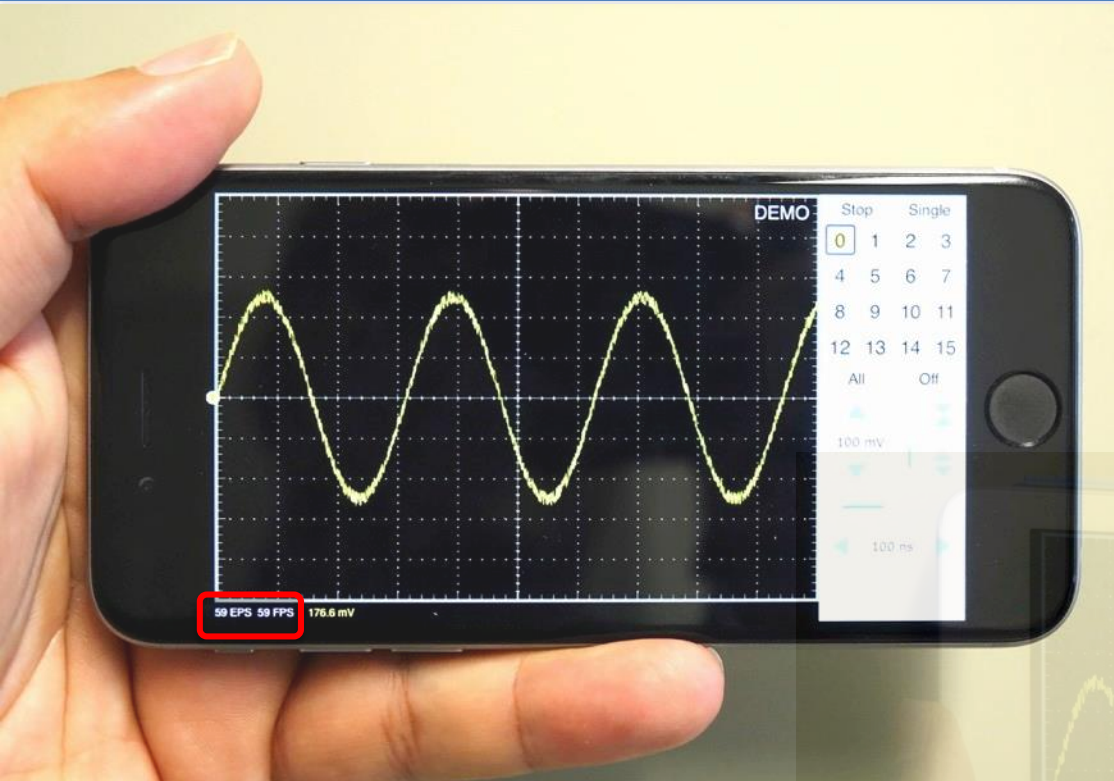
Client HV



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 on Indico