Avancerad webbprogrammering
sedan 1997

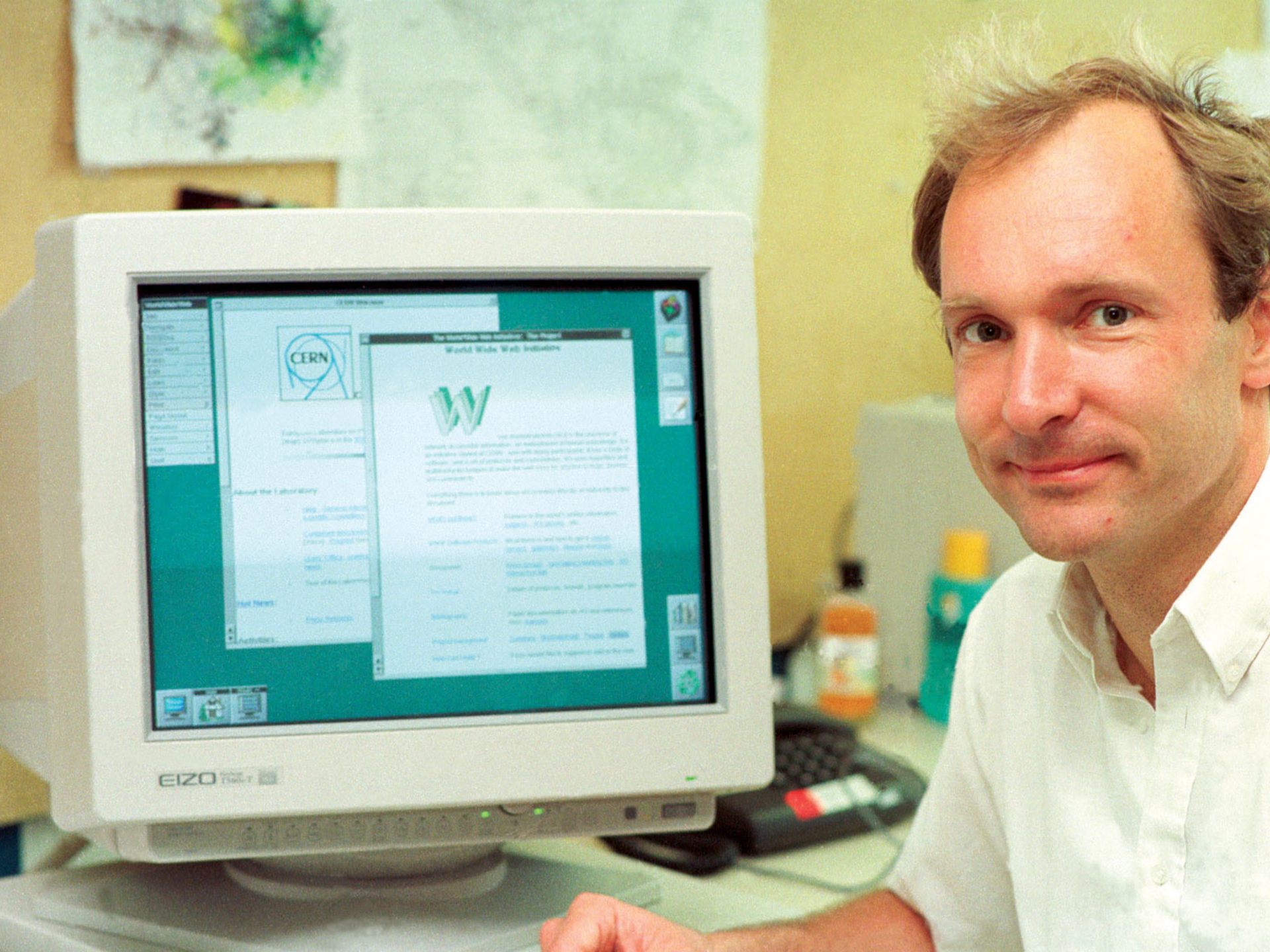
ERIK BERGLUND, LEKTOR

Webbapplikationsutveckling

Om webbutveckling

Rätt tänk om webb och applikationer

Rätt tänk för kursens tekniska plattform


CERN
World Wide Web Initiative
EIZO

Webbsida [redigera | redigera wikitext]

Ej att förväxla med Webbplats.

Webbsida är en enstaka sida på en webbplats.

Tekniskt är en webbsida en textfil uppbyggd av till exempel Hypertext Markup Language (HTML) eller Extensible Hypertext Markup Language (XHTML) som kan läsas av en webbläsare.

Det är möjligt för en webbsida att integrera olika typer av resurser. Dessa inkluderar stilinformation, som är ansvarig för hur sidan ser ut, skript som lägger till interaktion med en webbsida

Nomenklatur [

Ordet *webbsida* använd

Ordet *hemsida* (engelsk

”webbsida”, ”ingångssid

Se även [redigera |

- Hemsida
- Webbdesign
- Webbutveckling
- Webbhotell
- Webbplats

Källor [redigera | re

1. ^ What is the diffe
2. ^ ”hemsida”. Sv

- ”webbsida”. Svens

Kategori: World Wide

Webbapplikation [redigera | redigera wikitext]

(Omdirigerad från Webbapp)

Den här artikeln **behöver källhänvisningar för att kunna verifieras.** (2022-

Åtgärda genom att lägga till pålitliga källor (gärna som fotnoter). Uppgifter utan källhänvisning

En **webbapplikation** är ett samlingsnamn för programvara som man kommer åt genom att använda en webbläsare

Se även [redigera | redigera wikitext]

- AJAX
- Mobilapplikation
- Molntjänst
- Platform as a service
- Software as a service
- Tillämpningsprogram
- Tunn klient
- Rich Web Applications
- Webbtjänst

Tekniska krav

Följande tekniska krav ska gälla:

- Relevant optimering av laddningstider och nätverksprestanda.
- Webbapplikationen ska byggas med huvudsaklingen **Bootstrap**, **jQuery** (JavaScript), oc
- Data i webbapplikationen ska lagras i en databas.
- Utvecklingsprojektet ska fungera som en webbapplikation vilket innebär att så fort man g
  skillnaden (extra informationen/datat) hämtas (och inte en hel webbsida) och presentera
- Utvecklingsprojektet ska bestå av en back-end och en front-end som endast kommunicer
- Om webbapplikationen ska användas på olika enheter så ska den anpassas med avseend
  desktop (ev. även TV-stora). Bootstrap är byggt för responsiv design (som detta kallas) m
  lämpliga komponenter/lämplig design väljas.
- Webbapplikationen ska versionhanteras på gitlab.liu.se

- Visning av innehåll på ett för kunden/användaren relevant sätt (t.ex. produkte
- Insamling av relevant data t.ex. om kunder/användare, genomförda aktivitete professionell hantering av reklmationer)
- En utvecklad och professionell betalprocess med relevanta steg (t.ex. beställn e-butik) simulerad eller realiserad med t.ex. **Stripe** eller **PayPals** utvecklar-s **Express Checkout**, eftersom många andra integreringar inte fungerar i Sveri
- Möjlighet för användaren (kund och leverantör) att återgå till gamla händelse en reklamation).
- Om webbapplikationen ska editeras av en adminstratör då ska online-editerin

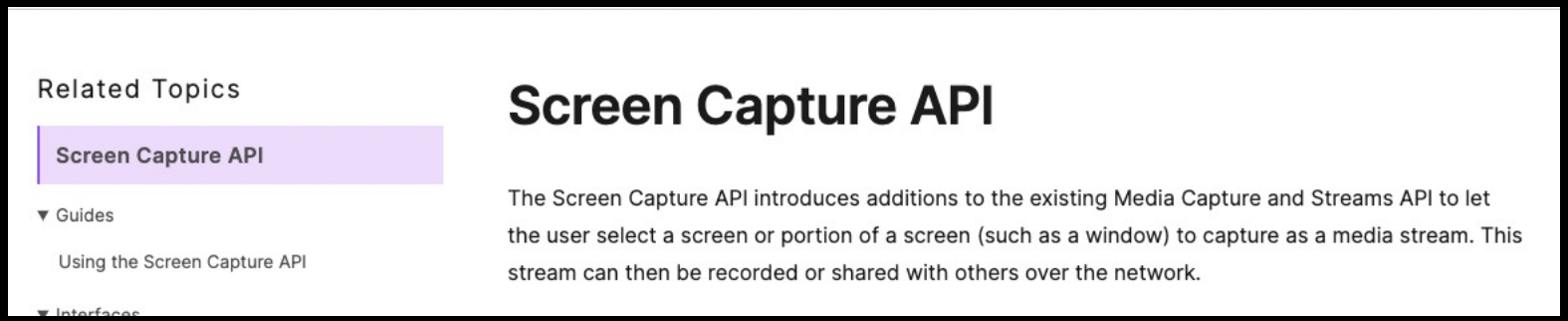
TIOBE Index for January 2023

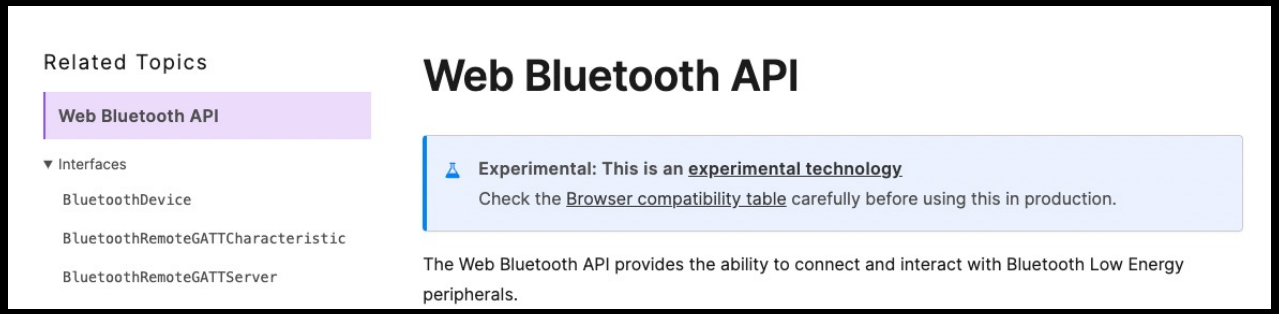
Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		Python	16.36%	+2.78%
2	2		C	16.26%	+3.82%
3	4	↑	C++	12.91%	+4.62%
4	3	↓	Java	12.21%	+1.55%
5	5		C#	5.73%	+0.05%
6	6		Visual Basic	4.64%	-0.10%
7	7		JavaScript	2.87%	+0.78%
8	9	↑	SQL	2.50%	+0.70%
9	8	↓	Assembly language	1.60%	-0.25%
10	11	↑	PHP	1.39%	-0.00%

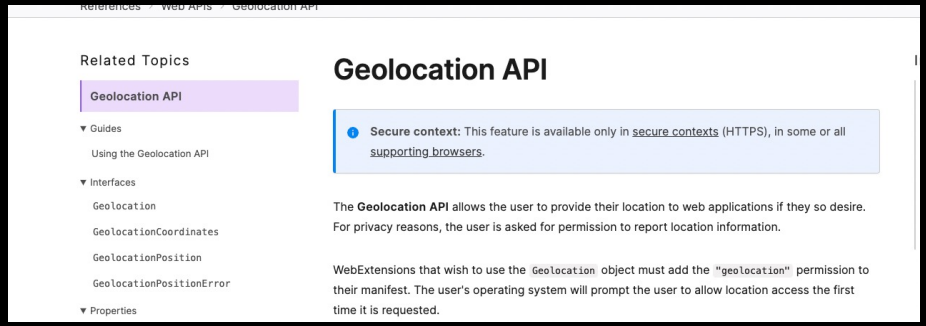
The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The operating system

- HTML, CSS, JavaScript
  - WASM, C++ compiled
  - JavaScript *Enkeltrådat
    - Web Workers kan göra beräknignar på andra trådar men inte full access
  - Kan anorpa servera för att hämta ny data
  - Parallella dataanrop och strömmar av data
- Rättighetssystem liknande mobil för åtkomst av fillager, webbkamera, mikrofon, notiser osv...
  - Särskillda Web APIer







Om Web API finns, enkelt att jobba med, annars mycket svårt att jobba med

Deployment

- Uppdatera system 30 gånger om dagen till användare!
- Ingen annan plattform tillåter så frekvent uppdatering
  - Bra för iterativ utveckling och ständig anpassning

Progressive Web Apps

Responsive

Secure

Independent of Connectivity

Bye Bye App Sto

Push Notificatio

Fast

Highly Discovera

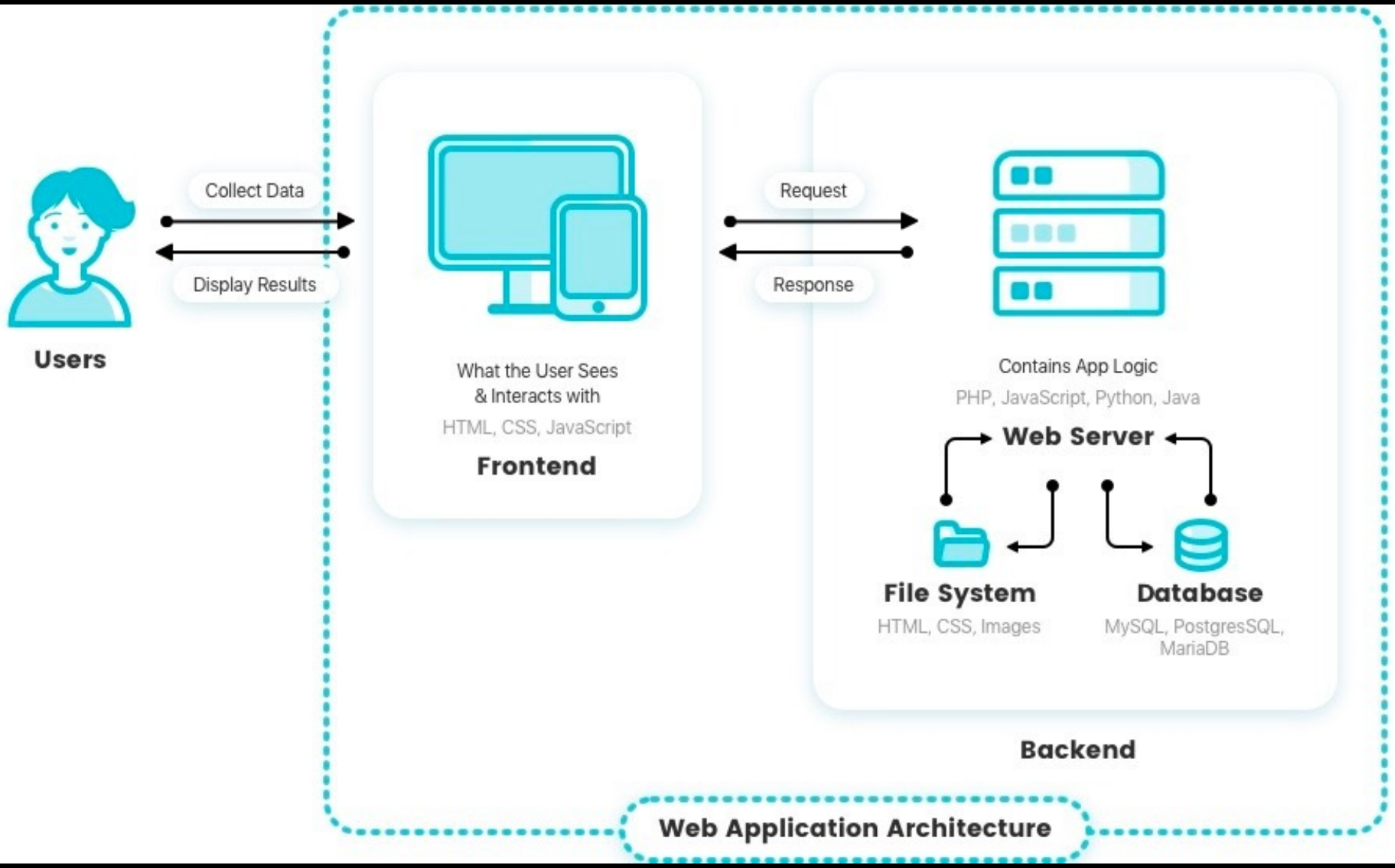
ELECTRON

Ej denna kurs, de som vill läser sådant i TDDD27 vt2 (om ett år?)

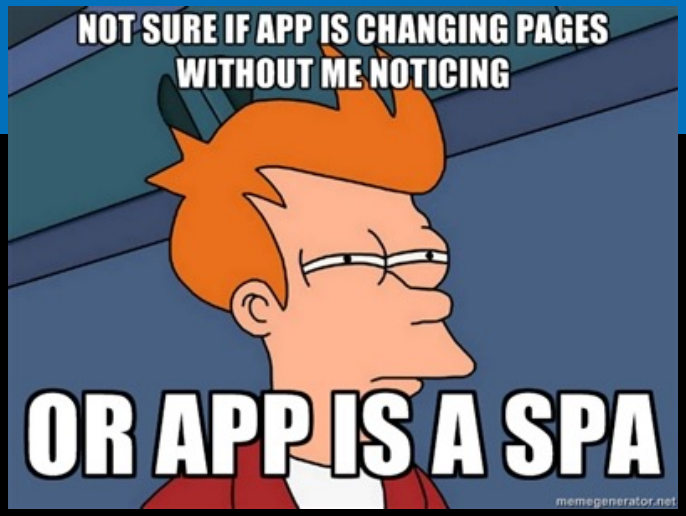
TypeScript vs. JavaScript:
The Difference You Should Know

RADIX

www.radixweb.com

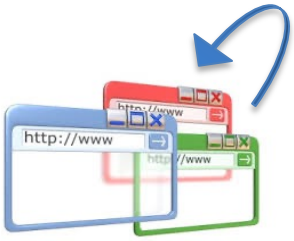

Collect Data
Display Results
Users
What the User Sees
& Interacts with
HTML, CSS, JavaScript
Frontend
Request
Response
Contains App Logic
PHP, JavaScript, Python, Java
Web Server
File System
HTML, CSS, Images
Database
MySQL, PostgresSQL,
MariaDB
Backend
Web Application Architecture

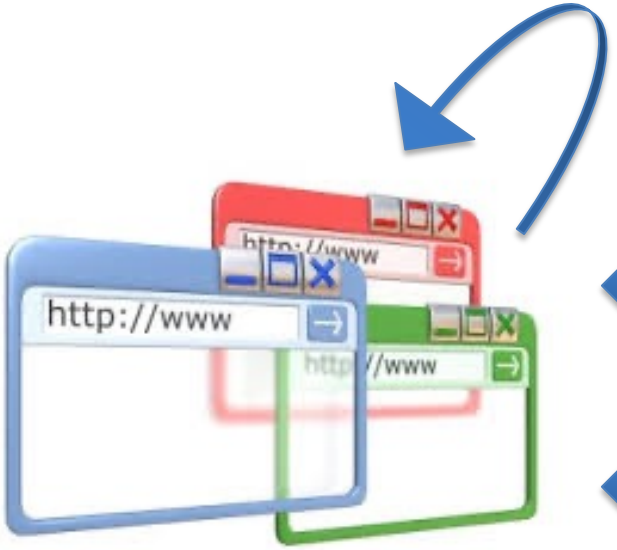
Ingen religion, men …



- När sidan drivs från klienten
  - Minimla kostnad
  - Maximal responsivitet i webläsaren
  - Maximalt antal samtidiga användare
- När sidan drivs från backend
  - Maximal upptid första gången
  - Svaga och gamla klient-maskiner fungerar bäst

webbapp kör i webbläsaren



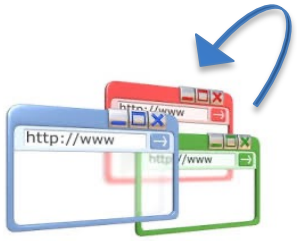


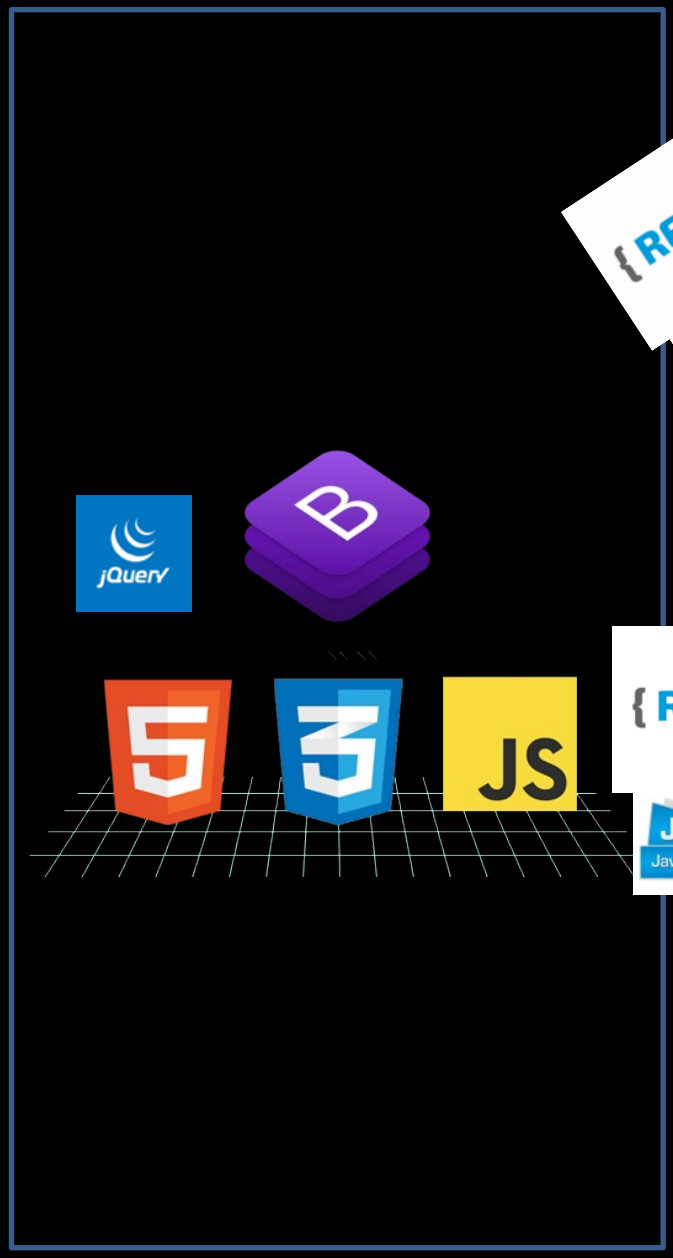
Installera/ ladda webbappen

HTML/CSS/JavaScript

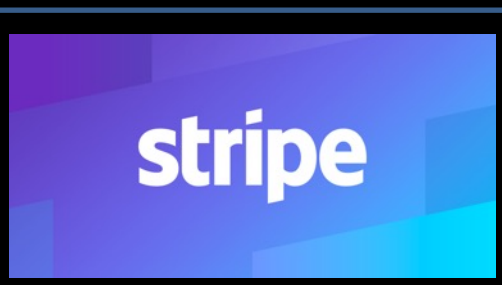
Hämta/skicka data mot server

JSON




jQuery
JS


{ REST:API }
JSON
JavaScript Object Notation


stripe


{ REST:API }
JSON


{ REST:API }
JSON
JavaScript Object Notation

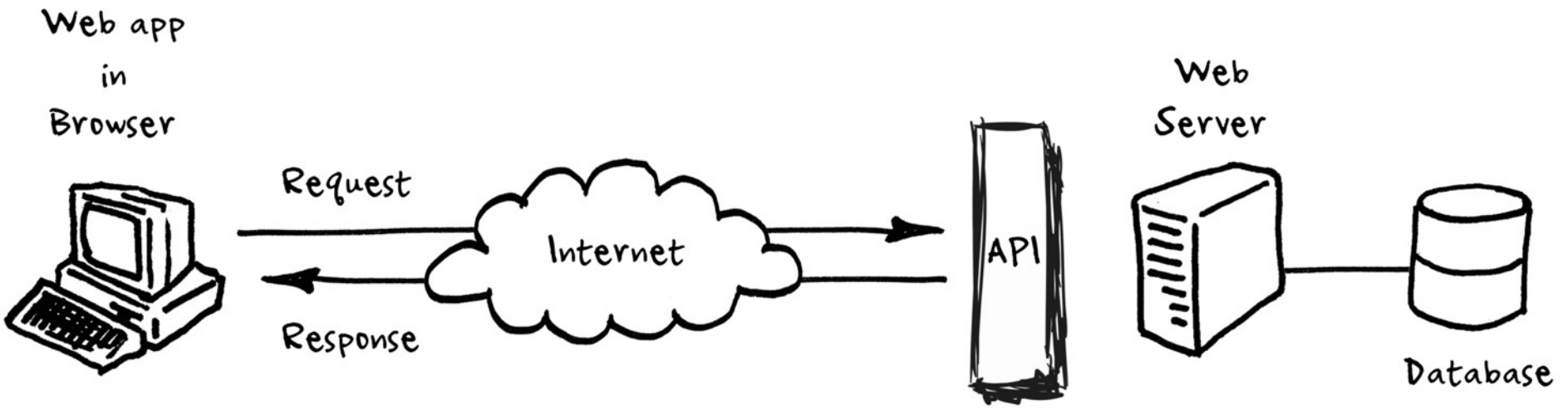

Flask
web development,
one drop at a time

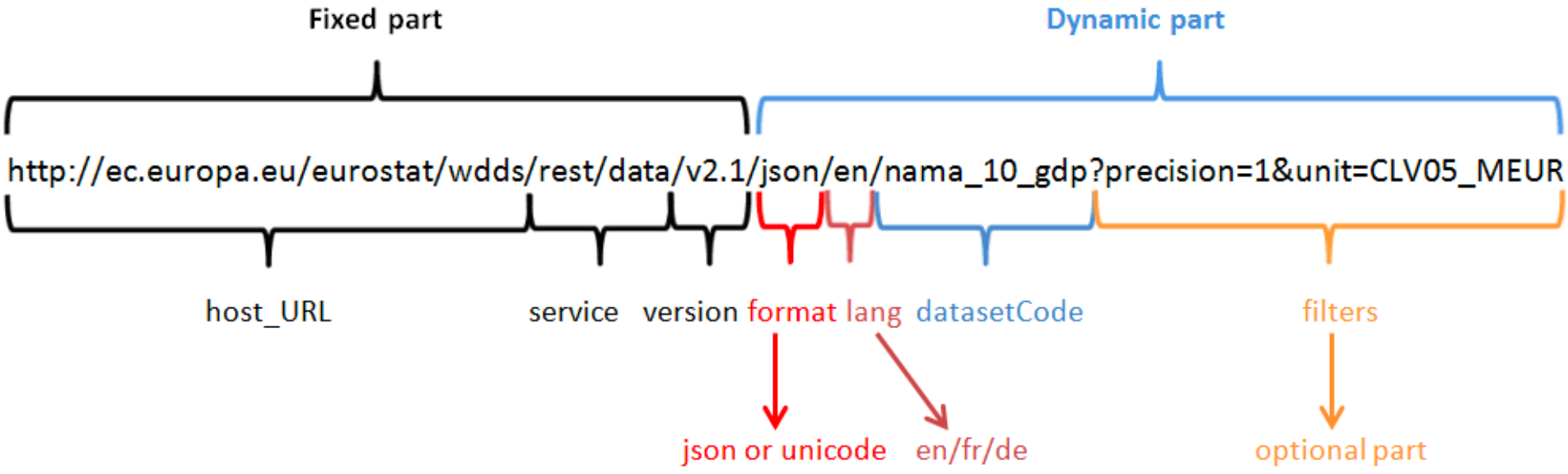
PYTHON

SQLite
VS
PostgreSQL
VS
MySQL

RESTful API

GET PUT POST DELETE





- Nielsen URL rules 1999 (URLs as UI):
  - a **domain name** that is easy to remember and easy to spell
  - **short** URLs
  - **easy-to-type** URLs
  - URLs that visualize the **site structure**
  - URLs that are "**hackable**" to allow users to move to higher levels of the information architecture by hacking off the end of the URL
  - **persistent URLs** that don't change

Webbens alla syntaxer

```
<!DOCTYPE>
<html>

<head>
<titile> Demo </title>
</head>

<body>
<p> this is first demo of html structure </p>
</body>

</html>
```

Identifiera grupp för stil och annat

Identifiera specifikt object

Document.getElementById()

Document.getElementsByClassName()

```
<html>
<head>
<style>
div.img {
    margin: 5px;
    padding: 5px;
    border: 1px solid #0000ff;
    height: auto;
    width: auto;
    float: left;
    text-align: center;
}
```

Example

Set the font size for different elements:

```
div.a {
  font-size: 15px;
}

div.b {
  font-size: large;
}

div.c {
  font-size: 150%;
}
```

Try it Yourself »

Relativ
Absoult
Fixed
...

tal
procent
stor, liten, mindre...
auto

	Recommended	Occasional use	Not recommended
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	

```
<head>
<style>
hr {color: red;}
p {margin-left: 20px;}
body {background-image: url("images/background.gif");}
</style>
</head>
```

Example

Change the background color of the <body> element to "lightblue" when the browser window is 600px wide or less:

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Även tal i CSS är text.
DVS vi kan inte direkt
räkna med positioner

```
function changeBackground(i) {
    var colors = ["red", "green", "blue", "purple", "orange"];
    if (i == colors.length) {
        i = 0;
    }
    color = colors[i];
    document.body.style.backgroundColor = color;
    console.log(i);
    return i;
}

function Increment() {
    changeBackground(i);
```

OBS!

```
<style>
    #d1 {
        display: none !important;
        color: red;
    }
</style>
<script>
    let showElement = () => {
        let ele = document.getElementById('d1');
        ele.setAttribute ('style', 'display: block !important;');
    }
</script>
```

Overriding this property

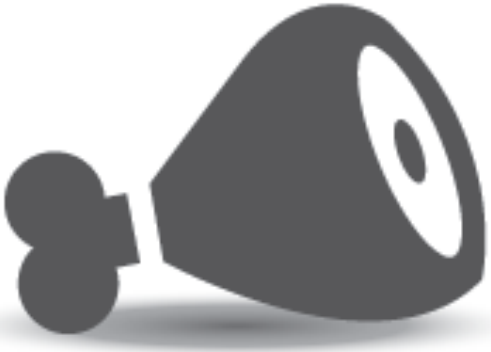
Now, this is important.

```
<html>
  <head>
    <div>
          <div>

              <form method="post" action="#" id="formvalue" onkeyup="
              drawChart()" />

              </form>
          </div>
    </div>



    <script type="text/javascript" src="https://www.google.com/jsapi"></
      script>
    <script type="text/javascript">

    var bid = 43;
    var ask = 21;


      google.load("visualization", "1", {packages:["corechart"]});
      google.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Price', 'Quantity'],
          ['Value #1', bid],
          ['Value #2', ask],
        ]);
```

ordning
spelar roll

JAVA *is to* JAVASCRIPT

as HAM *is to* HAMSTER



ILLUSTRATED BY SEGUE TECHNOLOGIES

```
41  function parse (i, body) {
42    var obj = {};
43    body.split('\n')
44    .forEach(v=>v.replace(/\s*(.*)\s*:\s*(.*)\s*/,
45             (s,key,val)=>{obj[key]=isNaN(val)||val.length<1?val||undefined:Number(val);
46                           }));
47
48    var objArr = Object.values(obj);
49
50    var res = [];
51    res[0] = i
52    res.push(objArr)
53
54    return res
55  }
```

Objects as first-class citizens

Functions as First-class citizens(supports functional programming)

Dynamic Typing

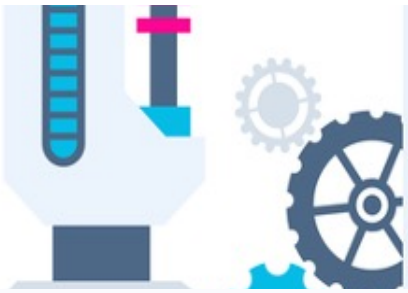
var dynamicType = “a string”
dynamicType = 5

```
let x = 1;

if (x === 1) {
  let x = 2;

  console.log(x);
  // Expected output: 2
}

console.log(x);
// Expected output: 1
```

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

```
const personPrototype = {
  greet() {
    console.log("hello!");
  },
};

const carl = Object.create(personPrototype);
carl.greet(); // hello!
```

JavaScript's Prototype



JAVASCRIPT

```
1 let wheels = { wheels: 4 }
2 let color = { color: "black"}
3 const Car = () => {
4     return Object.assign({}, wheels, color);
5 }
6 let car = Car();
7 console.log(car);
```

1. `false`, `0`, empty strings (`""`), `NaN`, `null`, and `undefined` all become `false`.
2. All other values become `true`.

Use <script async> when the script is independent(has no dependency). When the script depends, use <script defer>.

this

A function's **`this`** keyword behaves a little differently in JavaScript compared to other languages. It also has some differences between strict mode and non-strict mode.

In most cases, the value of `this` is determined by how a function is called (runtime binding). It can't be set by assignment during execution, and it may be different each time the function is called. The `bind()` method can set the value of a function's `this` regardless of how it's called, and arrow functions don't provide their own `this` binding (it retains the `this` value of the enclosing lexical context).

Callback Hell

A guide to writing asynchronous JavaScript programs

What is "*callback hell*"?

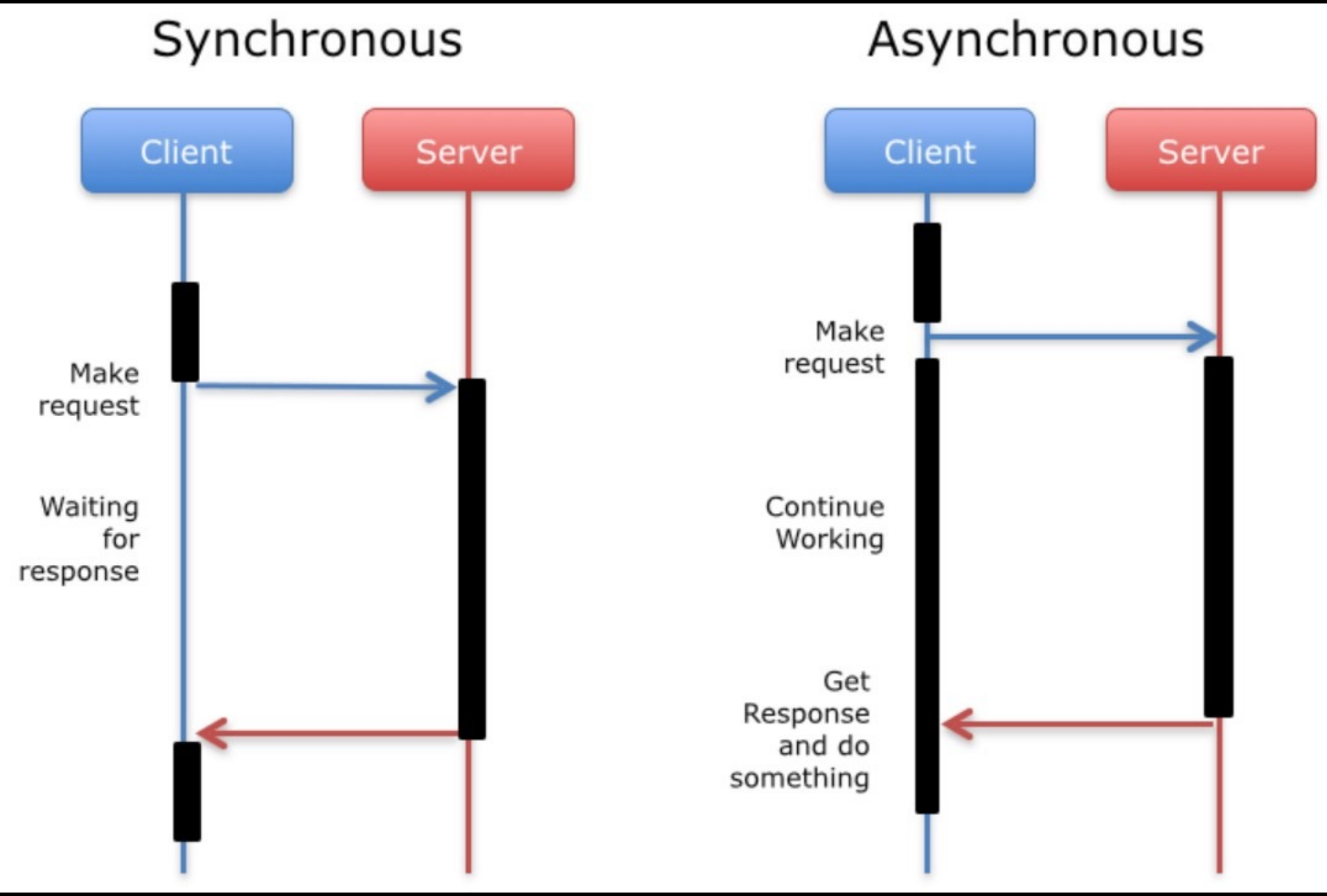
Asynchronous JavaScript, or JavaScript that uses callbacks, is hard to get right intuitively. A lot of code ends up looking like this:

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

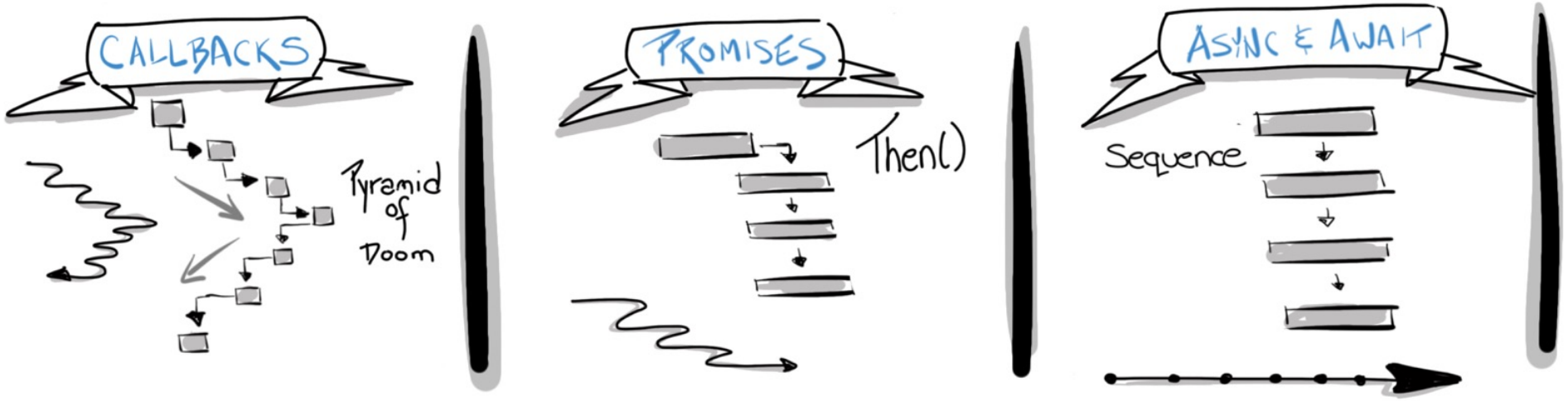
See the pyramid shape and all the `})` at the end? Eek! This is affectionately known as **callback hell**.

The cause of callback hell is when people try to write JavaScript in a way where execution happens visually from top to bottom. Lots of people make this mistake! In other languages like C, Ruby or Python there is the expectation that whatever happens on line 1 will finish before the code on line 2 starts running and so on down the file. As you will learn, JavaScript is different.

What are callbacks?


Synchronous
Client
Server
Make request
Waiting for response
Asynchronous
Client
Server
Make request
Continue Working
Get Response and do something

Asynchronous TypeScript



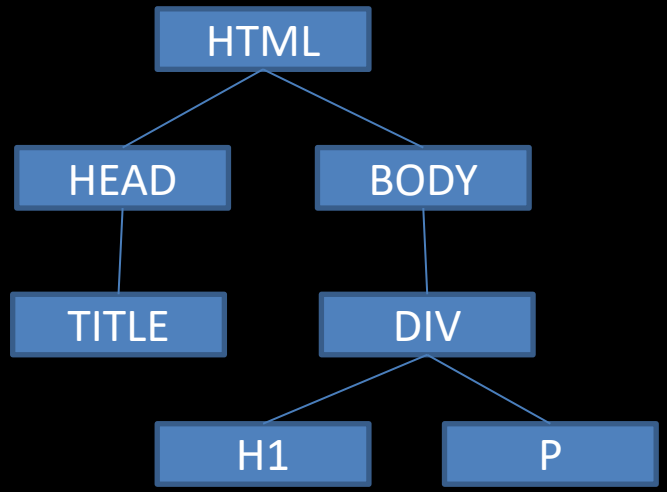
```
async function myFunc () {
    let val = await Promise
        .resolve(1)
        .then( x => x * 3)
        .then( x => x + 5)
        .then( x => x / 2);

    return val;
}
myFunc()
    .then( x => console.log( `x: ${x}` ) ); // 'x: 4'
```

HTML & DOM

- Document Object Model

```
<html>
    <head>
        <title>my title</title>
    </head>
    <body>
        <div>
            <h1>Headline</h1>
            <p>Hello world!</p>
        </div>
    </body>
</html>
```

Browser



HTML File
Code in file

DOM Tree
Data structure in memory
(Note: Each node of this tree can be access as a JavaScript Object)

JSONasJs

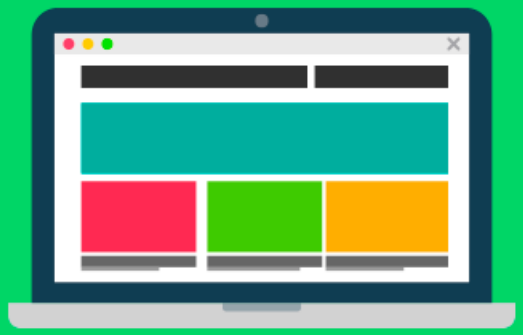
```
{menu: {
  id: "file",
  value: "File",
  popup: {
    menuitem: [
      {value: "New", onclick: "CreateNewDoc()"},
      {value: "Open", onclick: "OpenDoc()"},
      {value: "Close", onclick: "CloseDoc()"}
    ]
  }
}}
```

Show plain JSON

```
{
   status : 200 (number)
   error : [ »
      0 : 0 (number)
      1 : No errors (string)
   ]
   emptyObject : { *empty* }
   emptyArray : [ *empty* ]
   data : { »
      company_id : 268355140124
      country_id : 39 (string)
      timezone_id : 1 (string)
```

```
<script>
    var test = 'test-value';

    $('body').on('click', '#some_button', function () {
        var $this = $(this);
        $this.append('<i class="icon-spinner icon-spin"></i>');

        $.ajax({
            url: test,
            data: {
                test: 'test'
            }
        });
    });
</script>
```

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hey, we have Flask in a Docker container!'


if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```



FRONTEND

BACKEND

Home Docs Examples Icons Themes Blog

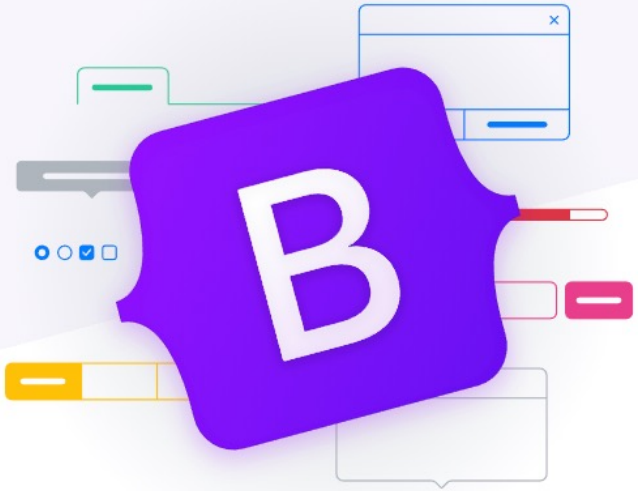
Download

Build fast, responsive sites with Bootstrap

Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

Get started Download

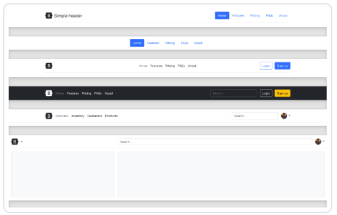
Currently **v5.0.0-beta1** · Previous releases



V4.6 (JQuery first)
V5 – beta (JavaScript first)

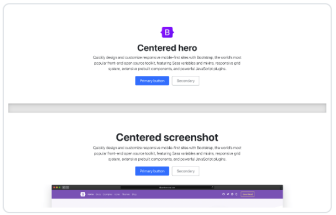
Snippets

Common patterns for building sites and apps that build on existing components and utilities with custom CSS and more.



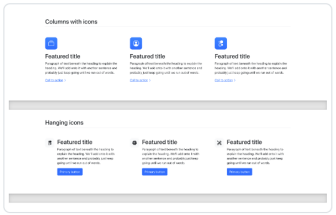
Headers

Display your branding, navigation, search, and more with these header components



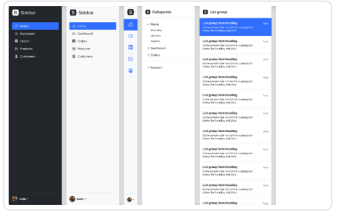
Heroes

Set the stage on your homepage with heroes that feature clear calls to action.



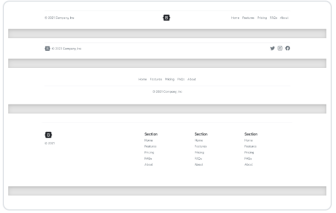
Features

Explain the features, benefits, or other details in your marketing content.



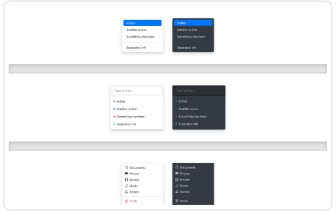
Sidebars

Common navigation patterns ideal for offcanvas or multi-column layouts.



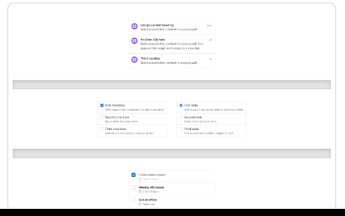
Footers

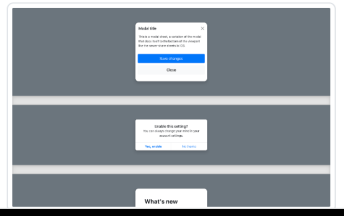
Finish every page strong with an awesome footer, big or small.

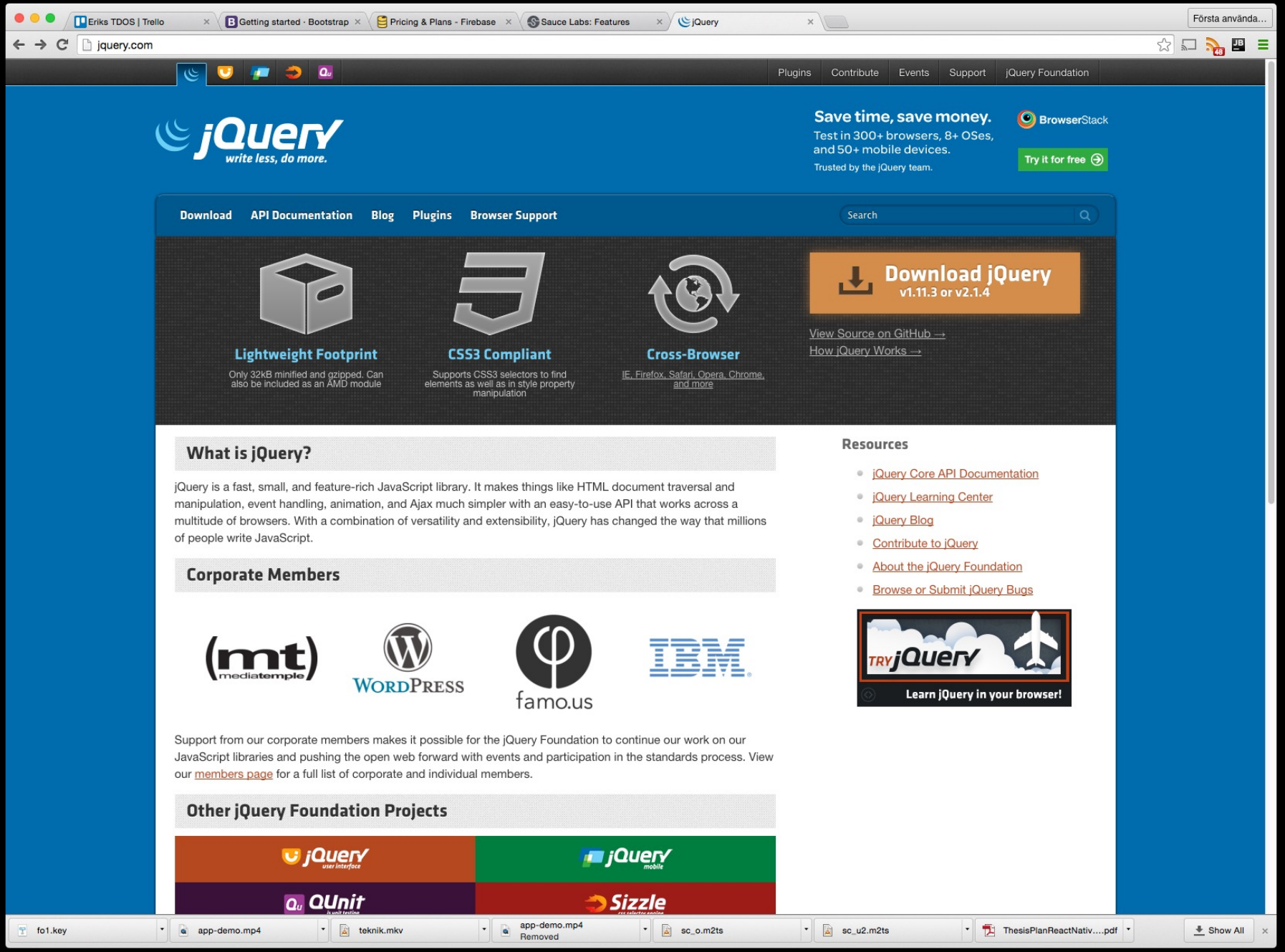


Dropdowns

Enhance your dropdowns with filters, icons, custom styles, and more.






Eriks TDOS | Trello
Getting started · Bootstrap
Pricing & Plans - Firebase
Sauce Labs: Features
jQuery
Första använda...
jquery.com
Plugins
Contribute
Events
Support
jQuery Foundation
jQuery
write less, do more.
Save time, save money.
Test in 300+ browsers, 8+ OSes, and 50+ mobile devices.
Trusted by the jQuery team.
BrowserStack
Try it for free
Download
API Documentation
Blog
Plugins
Browser Support
Search
Lightweight Footprint
Only 32kB minified and gzipped. Can also be included as an AMD module
CSS3 Compliant
Supports CSS3 selectors to find elements as well as in style property manipulation
Cross-Browser
IE, Firefox, Safari, Opera, Chrome, and more
Download jQuery
v1.11.3 or v2.1.4
View Source on GitHub →
How jQuery Works →
What is jQuery?
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.
Corporate Members
mediatemple
WordPress
famo.us
IBM
Support from our corporate members makes it possible for the jQuery Foundation to continue our work on our JavaScript libraries and pushing the open web forward with events and participation in the standards process. View our members page for a full list of corporate and individual members.
Other jQuery Foundation Projects
jQuery user interface
jQuery mobile
QUnit
Sizzle
Resources
jQuery Core API Documentation
jQuery Learning Center
jQuery Blog
Contribute to jQuery
About the jQuery Foundation
Browse or Submit jQuery Bugs
TRY jQuery
Learn jQuery in your browser!
fo1.key
app-demo.mp4
teknik.mkv
app-demo.mp4
Removed
sc_o.m2ts
sc_u2.m2ts
ThesisPlanReactNativ....pdf
Show All

DOM Traversal and Manipulation

Get the `<button>` element with the class 'continue' and change its HTML to 'Next Step...'

```
$( "button.continue" ).html( "Next Step..." )
```

Event Handling

Show the `#banner-message` element that is hidden with `display:none` in its CSS when any button in `#button-container` is clicked.

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
  hiddenBox.show();
});
```

Ajax

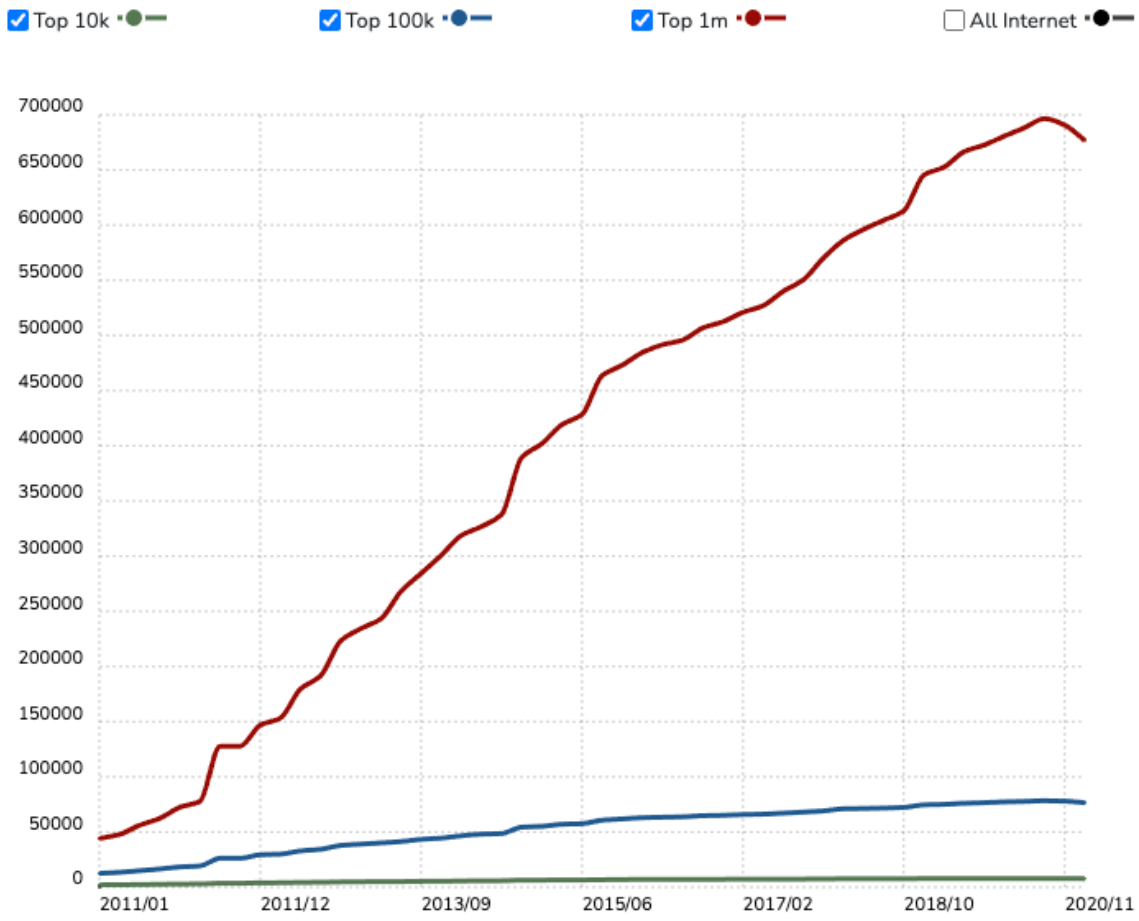
Call a local script on the server `/api/getWeather` with the query parameter `zipcode=97201` and replace the element `#weather-temp`'s html with the returned tex

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $( "#weather-temp" ).html( "<strong>" + result + "</strong> degrees" );
  }
});
```

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

jQuery is a **JavaScript library** designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax.

jQuery Usage Statistics



Download Lead List

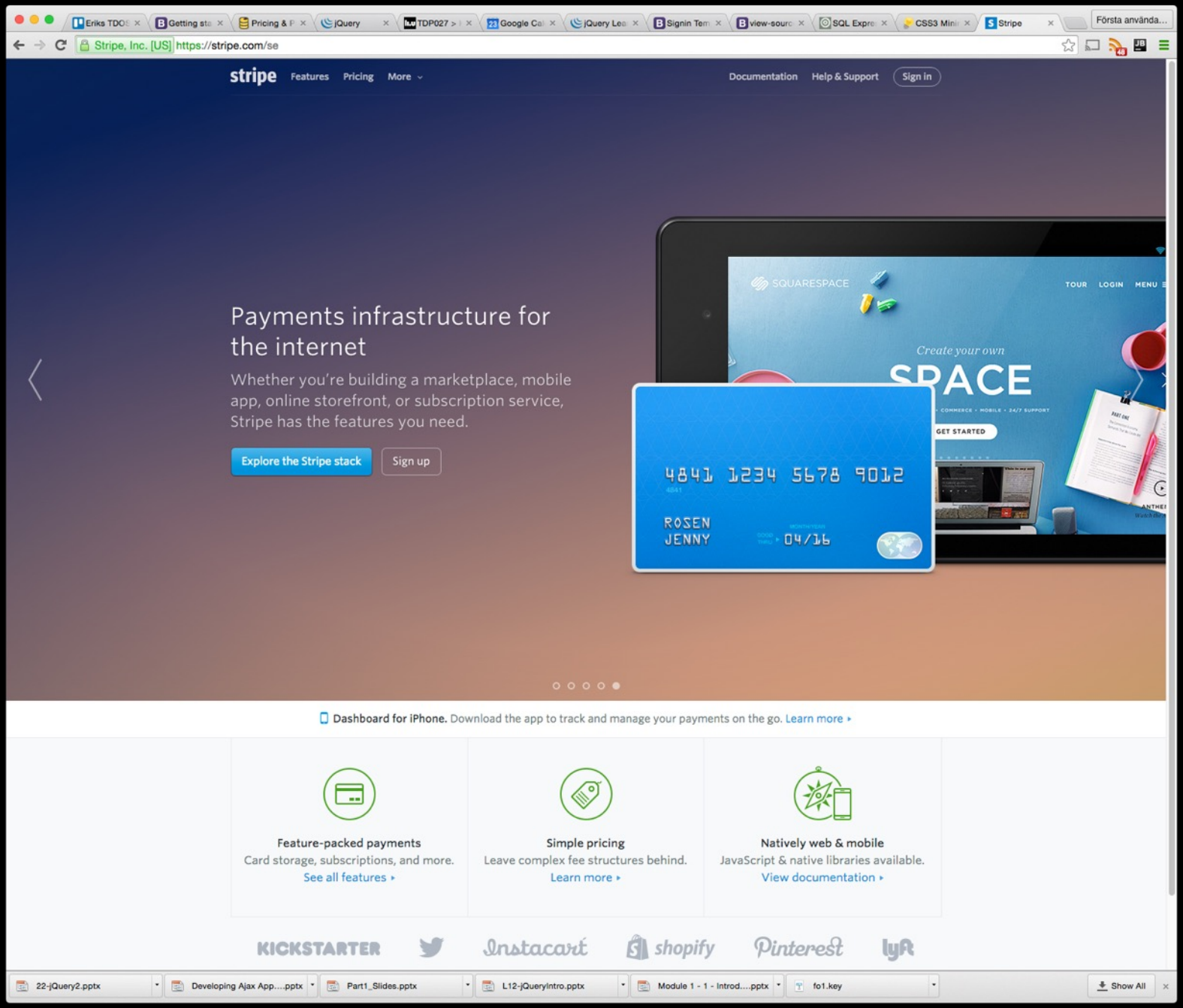
We know of at least 71,803,422 live websites using jQuery.

Site Totals

Total Live	71,803,422
4,992,521 additional website redirects[7].	
Swedish Live Sites	267,314 Estimated
Top 1m	73.83% 738,305
Top 100k	80.34% 80,339
Top 10k	79.81% 7,981
United States	41,862,215*
United Kingdom	3,133,706*
Russia	2,877,676*
Germany	2,173,899*
China	2,156,359*
Japan	1,554,851*

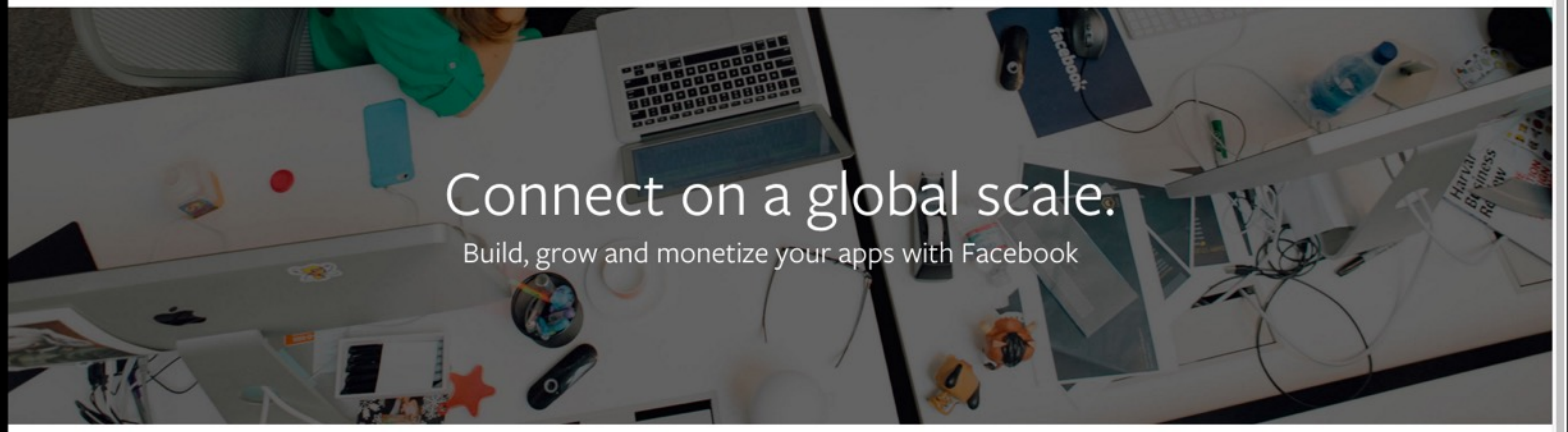
RESPONSIVE

WEB DESIGN


stripe
Features
Pricing
More
Documentation
Help & Support
Sign in
Payments infrastructure for the internet
Whether you're building a marketplace, mobile app, online storefront, or subscription service, Stripe has the features you need.
Explore the Stripe stack
Sign up
4841 1234 5678 9012
ROSEN JENNY
04/16
SQUARESPACE
TOUR
LOGIN
MENU
Create your own
SPACE
GET STARTED
Dashboard for iPhone. Download the app to track and manage your payments on the go. Learn more
Feature-packed payments
Card storage, subscriptions, and more.
See all features
Simple pricing
Leave complex fee structures behind.
Learn more
Natively web & mobile
JavaScript & native libraries available.
View documentation
KICKSTARTER
Instacart
shopify
Pinterest
lyft
22-jQuery2.pptx
Developing Ajax App....pptx
Part1_Slides.pptx
L12-jQueryIntro.pptx
Module 1 - 1 - Introd....pptx
fo1.key
Show All

https://developers.facebook.com

facebook for developers | Products | Docs | Tools & Support | News | Search | My Apps



Facebook Login
Account Creation in two taps

Sharing on Facebook
Promote your app or website organically

Facebook Social Plugins
Easily make your app or website social

Facebook Analytics for Apps
Understand how people use your app

App Monetization
Monetize with ads and publisher tools

View All Products

22-jQuery2.pptx | Developing Ajax App....pptx | Part1_Slides.pptx | L12-jQueryIntro.pptx | Module 1 - 1 - Introd....pptx | fo1.key | Show All

FRONTEND

BACKEND

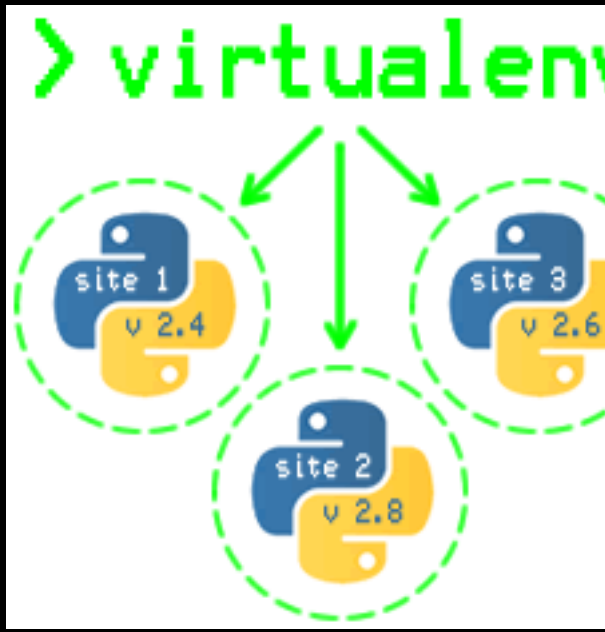
Flask

web development,
one drop at a time

python

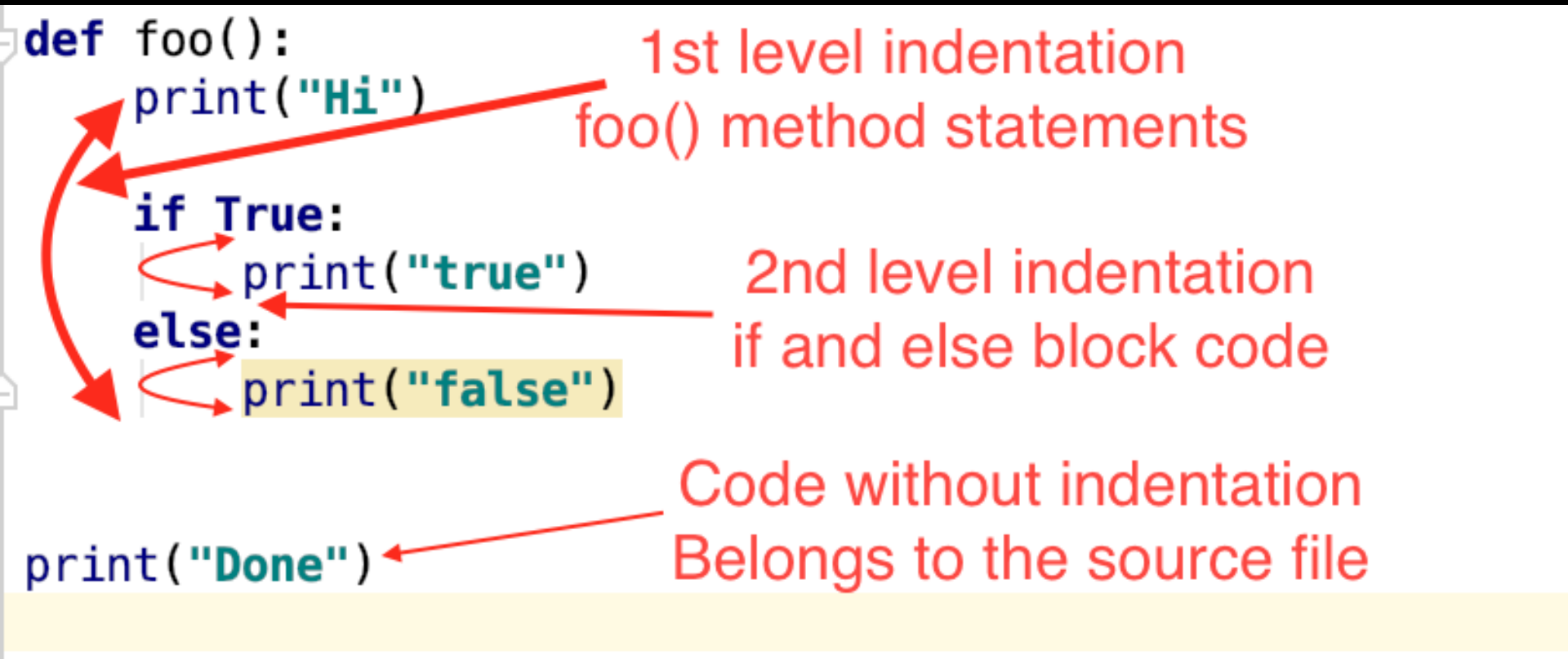
Flask
web development,
one drop at a time

Python



- Dynamisk typing “duck typing”
- TAB istället för {}
- No ; vid slut av uttryck

- Effektivt, enkelt, kortfattat
- Väldigt avancerad listhantering på språklig/grundläggande nivå


def foo():
print("Hi")
if True:
print("true")
else:
print("false")
print("Done")
1st level indentation
foo() method statements
2nd level indentation
if and else block code
Code without indentation
Belongs to the source file

Micro FRAMEWORKS

Slim

PHP Micro Framework

HOME WHY? README DOWNLOAD SOURCE CONTACT

SPARK

MICRO WEB FRAMEWORK

Flask

web development,
one drop at a time

NETFLIX

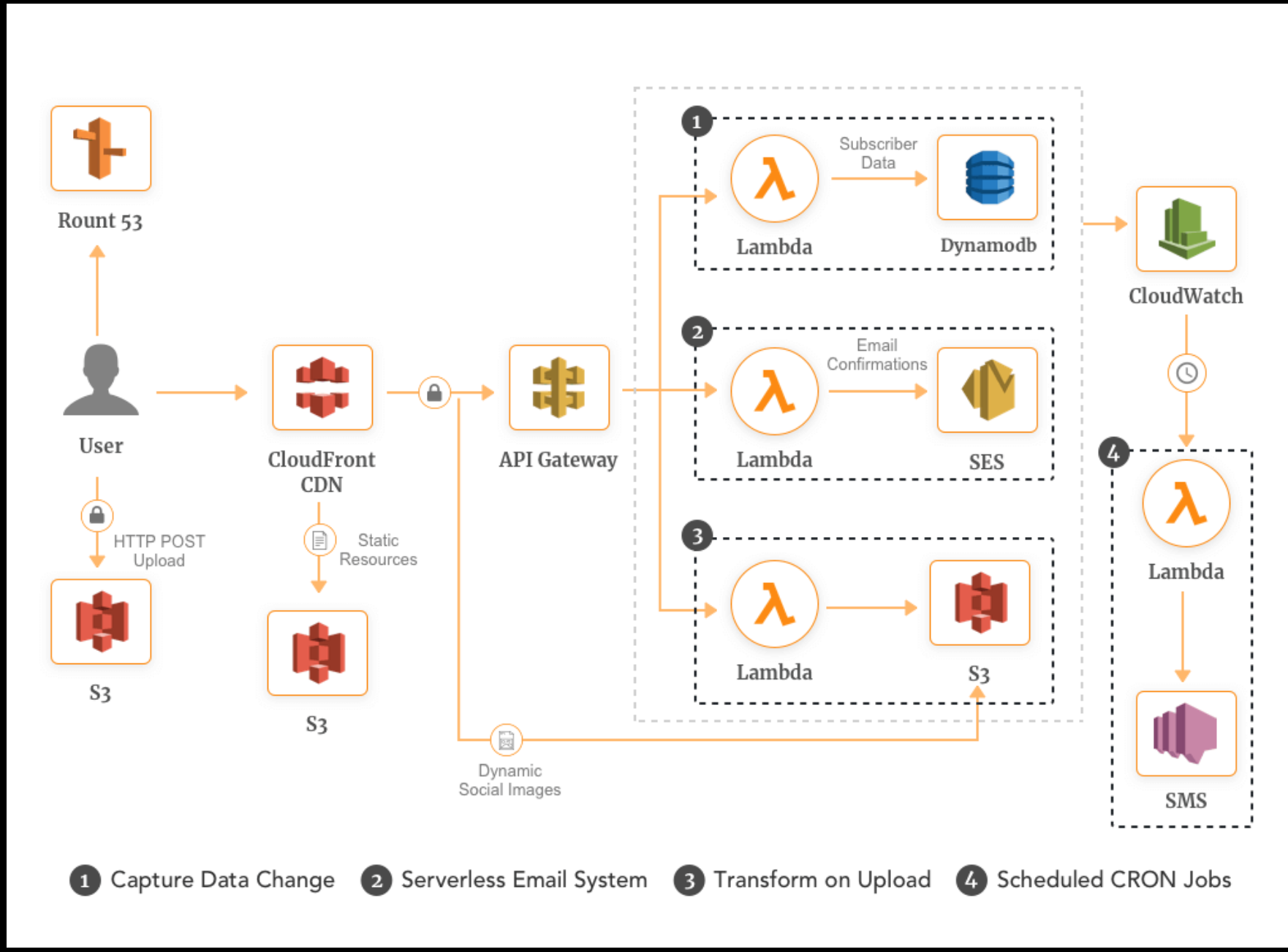
Scalatra

Express JS

microjs

Amazon Lambda vs Azure Functions:

Serverless Computing


Rount 53
User
S3
HTTP POST
Upload
CloudFront
CDN
Static
Resources
S3
API Gateway
Dynamic
Social Images
1
Lambda
Subscriber
Data
Dynamodb
2
Lambda
Email
Confirmations
SES
3
Lambda
S3
CloudWatch
4
Lambda
SMS
1 Capture Data Change
2 Serverless Email System
3 Transform on Upload
4 Scheduled CRON Jobs

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

```python
from markupsafe import escape

@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % escape(username)

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return 'Subpath %s' % escape(subpath)
```

Run with a Production Server

When running publicly rather than in development, you should not use the built-in development server (`flask run`). The development server is provided by Werkzeug for convenience, but is not designed to be particularly efficient, stable, or secure.

Instead, use a production WSGI server. For example, to use Waitress, first install it in the virtual environment:

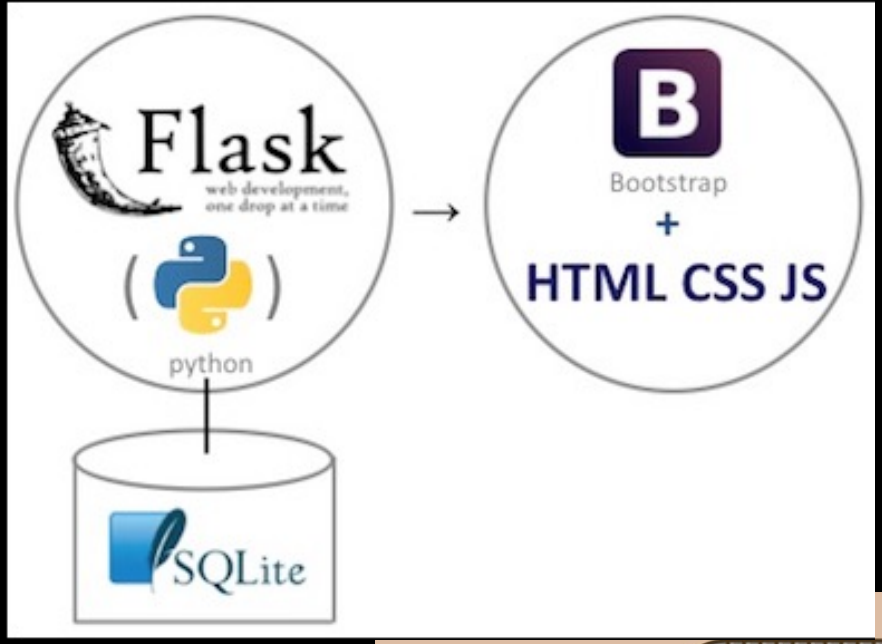
```
$ pip install waitress
```

You need to tell Waitress about your application, but it doesn't use `FLASK_APP` like `flask run` does. You need to tell it to import and call the application factory to get an application object.

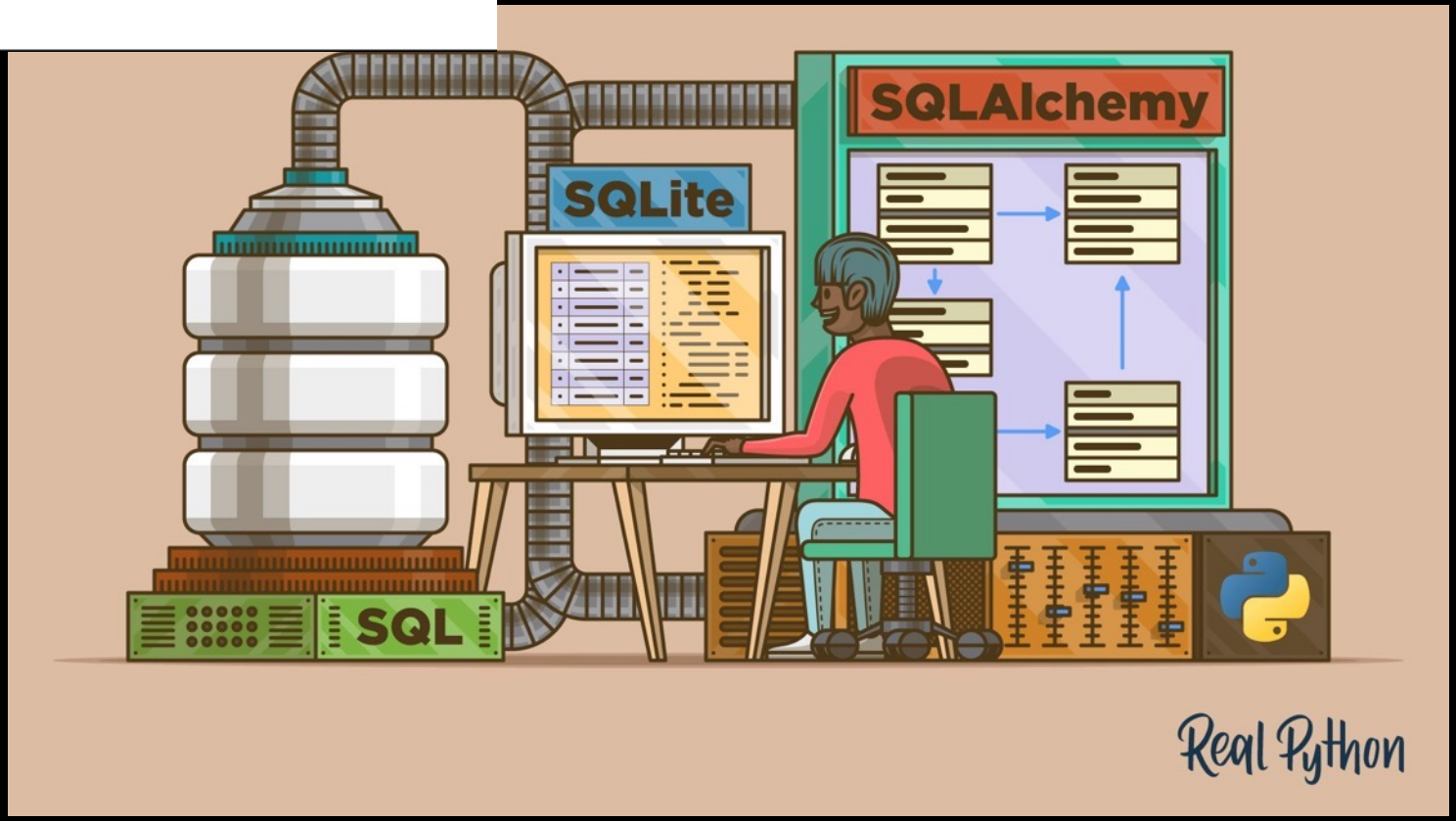
```
$ waitress-serve --call 'flaskr:create_app'

Serving on http://0.0.0.0:8080
```

See Deployment Options for a list of many different ways to host your application. Waitress is just an example, chosen for the tutorial because it supports both Windows and Linux. There are many more WSGI servers and deployment options that you may choose for your project.

Continue to Keep Developing!.



.gitignore sqlite-db in project



```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/get-json')
def hello():
    return jsonify(hello='world') # Returns HTTP Respo
```

It's a wrap for **Chrome Dev Summit 2020**! Watch all the sessions at goo.gle/cds20-sessions now!

Chrome DevTools

Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster.

Check out the video for live demonstrations of core DevTools workflows, including debugging CSS, prototyping CSS, debugging JavaScript, and analyzing load performance.

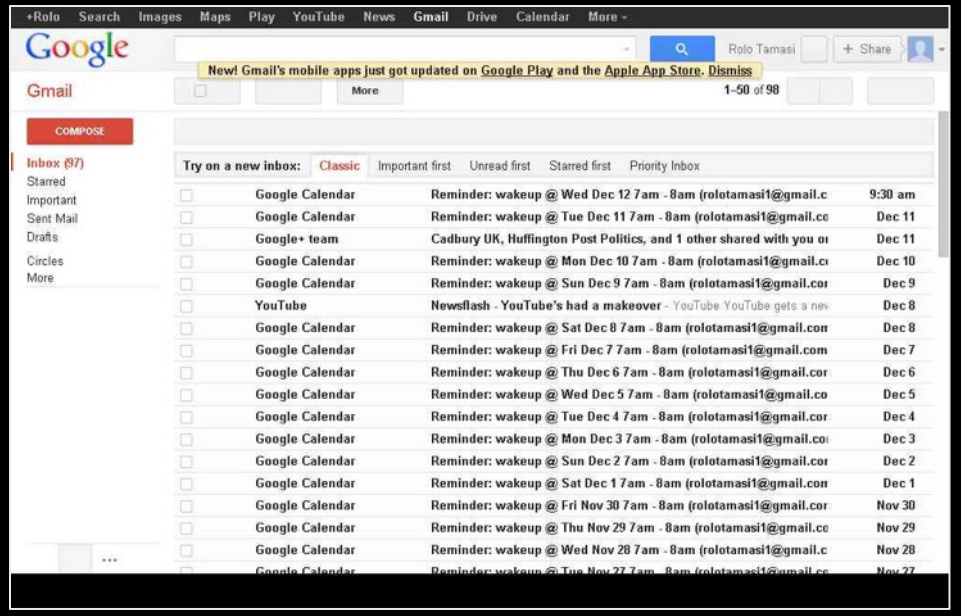
Open DevTools

There are many ways to open DevTools, because different users want quick access to different parts of the DevTools UI.

- When you want to work with the DOM or CSS, right-click an element on the page and select **Inspect** to jump into the **Elements** panel. Or press `Command+Option+C` (Mac) or `Control+Shift+C` (Windows, Linux, Chrome OS).
- When you want to see logged messages or run JavaScript, press `Command+Option+J` (Mac) or

3.3 seconds
window.onload

4.8 seconds
90% rendered



window.onload

≠

user perception

2.0 seconds
88% rendered

5.2 seconds
window.onload

KYLE RUSH

HOME ABOUT CONTACT

Meet the Obama campaign's $250 million fundraising platform

We made the new platform 60% faster and this resulted in a 14% increase in donation conversions.

- 6 month life span
- $250 million dollars, 4,276,463 donations
- 81,548,259 pageviews, 17,807,917 unique visitors
- 60% faster time to paint than previous platform
- 240 a/b tests, 49% increase

kylerush.net/blog/meet-the-obama-campaigns-250-million-fundraising-platform/

JavaScript Best Practices

❮ Previous

Next ❯

Avoid global variables, avoid `new`, avoid `==`, avoid `eval()`

Avoid Global Variables

Minimize the use of global variables.

This includes all data types, objects, and functions.

Global variables and functions can be overwritten by other scripts.

Use local variables instead, and learn how to use closures.

Always Declare Local Variables

All variables used in a function should be declared as **local** variables.

Local variables **must** be declared with the `var`, the `let`, or the `const` keyword, otherwise they will become global variables.

Initialize Variables

It is a good coding practice to initialize variables when you declare them.

This will:

- Give cleaner code
- Provide a single place to initialize variables
- Avoid undefined values

```
// Declare and initiate at the beginning
let firstName = "";
let lastName = "";
let price = 0;
let discount = 0;
let fullPrice = 0,
const myArray = [];
const myObject = {};
```

Declare Objects with **const**

Declaring objects with const will prevent any accidental change of type:

Example

```
let car = {type:"Fiat", model:"500", color:"white"};
car = "Fiat";      // Changes object to string
```

```
const car = {type:"Fiat", model:"500", color:"white"};
car = "Fiat";      // Not possible
```

Declare Arrays with **const**

Declaring arrays with const will prevent any accidential change of

Example

```
let cars = ["Saab", "Volvo", "BMW"];
cars = 3;    // Changes array to number
```

```
const cars = ["Saab", "Volvo", "BMW"];
cars = 3;    // Not possible
```

Don't Use new Object()

- Use `""` instead of `new String()`
- Use `0` instead of `new Number()`
- Use `false` instead of `new Boolean()`
- Use `{}` instead of `new Object()`
- Use `[]` instead of `new Array()`
- Use `/()/` instead of `new RegExp()`
- Use `function (){}` instead of `new Function()`

Use === Comparison

The `==` comparison operator always converts (to matching types) before comparison.

The `===` operator forces comparison of values and type:

Example

```
0 == "";        // true
1 == "1";       // true
1 == true;      // true

0 === "";       // false
1 === "1";      // false
1 === true;     // false
```

Try it Yourself »

Use Parameter Defaults

If a function is called with a missing argument, the value of the missing argument is set to `undefined`.

Undefined values can break your code. It is a good habit to assign default values to arguments.

Example

```
function myFunction(x, y) {
  if (y === undefined) {
    y = 0;
  }
}
```

Try it Yourself »

ECMAScript 2015 allows default parameters in the function definition:

```
function (a=1, b=1) { /*function code*/ }
```

Avoid Number, String, and Boolean as Objects

Always treat numbers, strings, or booleans as primitive values. Not as objects.

Declaring these types as objects, slows down execution speed, and produces nasty side effects:

Example

```
let x = "John";
let y = new String("John");
(x === y) // is false because x is a string and y is an object.
```

Try it Yourself »

Or even worse:

Example

```
let x = new String("John");
let y = new String("John");
(x == y) // is false because you cannot compare objects.
```

Try it Yourself »

JavaScript Common Mistakes

❮ Previous

Next ❯

This chapter points out some common JavaScript mistakes.

Accidentally Using the Assignment Operator

This `if` statement returns `true` (maybe not as expected), because 10 is true:

```
let x = 0;
if (x = 10)
```

Try it Yourself »

This `if` statement returns `false` (maybe not as expected), because 0 is false:

```
let x = 0;
if (x = 0)
```

Try it Yourself »

Expecting Loose Comparison

In regular comparison, data type does not matter. This `if` statement returns true:

```
let x = 10;
let y = "10";
if (x == y)
```

Try it Yourself »

In strict comparison, data type does matter. This `if` statement returns false:

```
let x = 10;
let y = "10";
if (x === y)
```

Try it Yourself »

Misunderstanding Floats

All numbers in JavaScript are stored as 64-bits **Floating point numbers** (Floats).

All programming languages, including JavaScript, have difficulties with precise floating point values:

```
let x = 0.1;
let y = 0.2;
let z = x + y            // the result in z will not be 0.3
```

Try it Yourself »

To solve the problem above, it helps to multiply and divide:

Example

```
let z = (x * 10 + y * 10) / 10;       // z will be 0.3
```

Try it Yourself »

All programming languages, including JavaScript, have difficulties with precise floating point values:

0.30000000000000004

Misplacing Semicolon

Because of a misplaced semicolon, this code block will execute regardless of the value of x:

```
if (x == 19);
{
  // code block
}
```

Try it Yourself »

Breaking a Return Statement

It is a default JavaScript behavior to close a statement automatically at the end of a line.

But, what will happen if you break the return statement in two lines like this:

Example 4

```
function myFunction(a) {
  let
  power = 10;
  return
  a * power;
}
```

Try it Yourself »

The function will return `undefined`!

Why? Because JavaScript thought you meant:

Accessing Arrays with Named Indexes

Example:

```
const person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
person.length;       // person.length will return 0
person[0];           // person[0] will return undefined
```

Undefined is Not Null

JavaScript objects, variables, properties, and methods can be `undefined`.

In addition, empty JavaScript objects can have the value `null`.

Correct:

```
if (typeof myObj !== "undefined" && myObj !== null)
```

Try it Yourself »