

tt()

Schedule

1. Events
2. Modules (import and export)
3. Async data fetching
4. All together!



Schedule

1. Events

2. Modules (import and export)

3. Async data fetching


4. All together!




Events

1. Mouse events (clicks, mouseover etc.)
2. Keyboard events (keypress etc.)
3. Touch events (longpress etc.)
4. Drag & Zoom events (drag etc.)

Events (transitions)


 D3

API index


Examples 


Visualization


d3-axis


d3-chord 


d3-color


d3-interpolate 

d3-contour 

d3-delaunay 

d3-force 


d3-geo 


d3-hierarchy 


d3-path


d3-polygon

d3-quadtree

d3-scale 

d3-scale-chromatic 



d3-selection 


d3-shape 

Animation

d3-ease

d3-time

 Search  K

7.9.0 GitHub 108.7k ★ Made by Observable 

d3-transition

A transition is a [selection](#)-like interface for animating changes to the DOM. Instead of applying changes instantaneously, transitions smoothly interpolate the DOM from its current state to the desired target state over a given duration.

To apply a transition, select elements, call [selection.transition](#), and then make the desired changes. For example:

```
d3.select("body")
  .transition()
  .style("background-color", "red");
```

Transitions support most selection methods (such as [transition.attr](#) and [transition.style](#) in place of [selection.attr](#) and [selection.style](#)), but not all methods are supported; for example, you must [append](#) elements or [bind data](#) before a transition starts. A [transition.remove](#) operator is provided for convenient removal of elements when the transition ends.

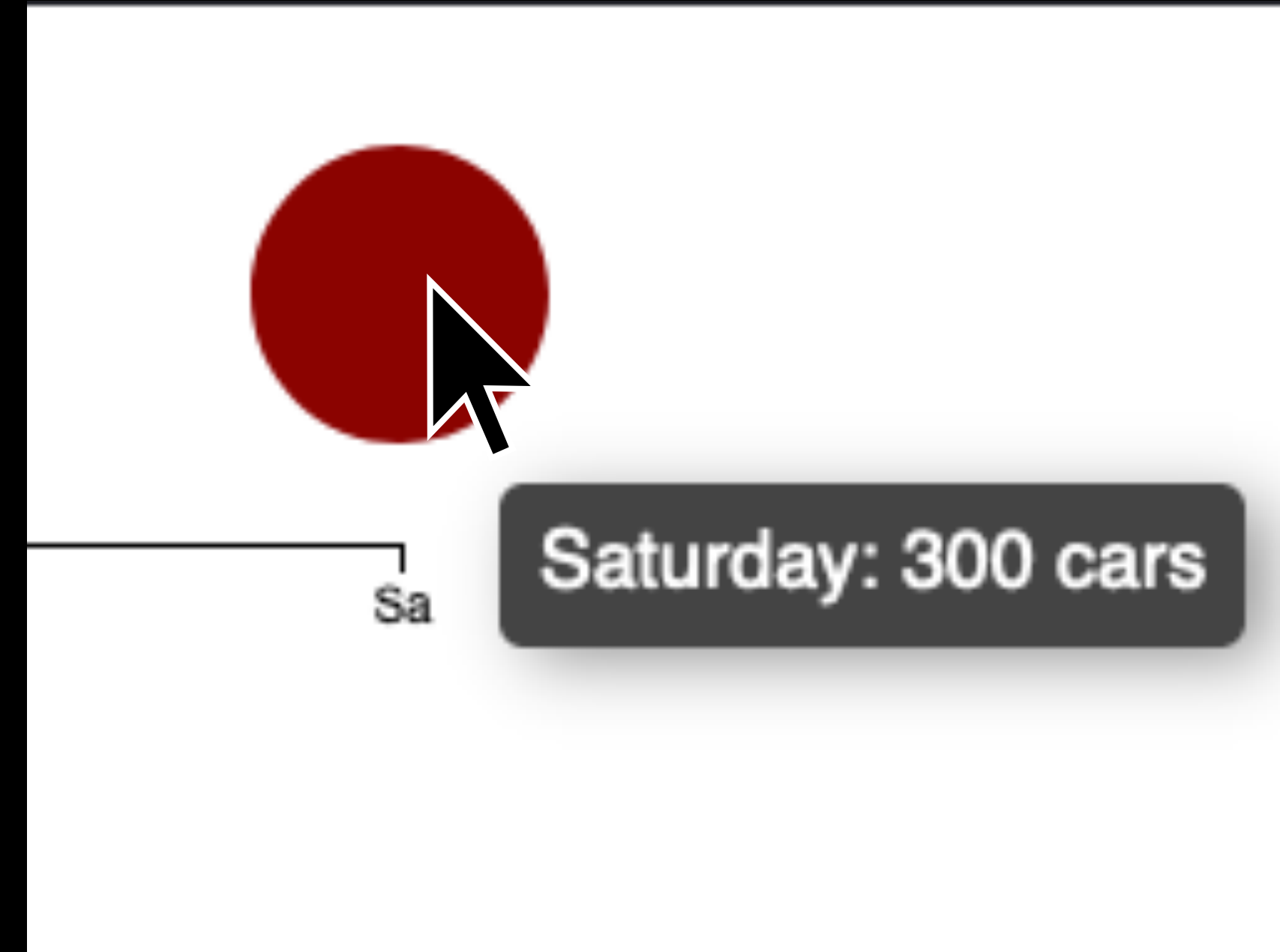
To compute intermediate state, transitions leverage a variety of [built-in interpolators](#). [Colors](#), [numbers](#), and [transforms](#) are automatically detected. [Strings](#) with embedded numbers are also detected, as is common with many styles (such as padding or font sizes) and paths. To specify a custom interpolator, use [transition.attrTween](#), [transition.styleTween](#) or [transition.tween](#).

See one of:

- [Selecting elements](#)

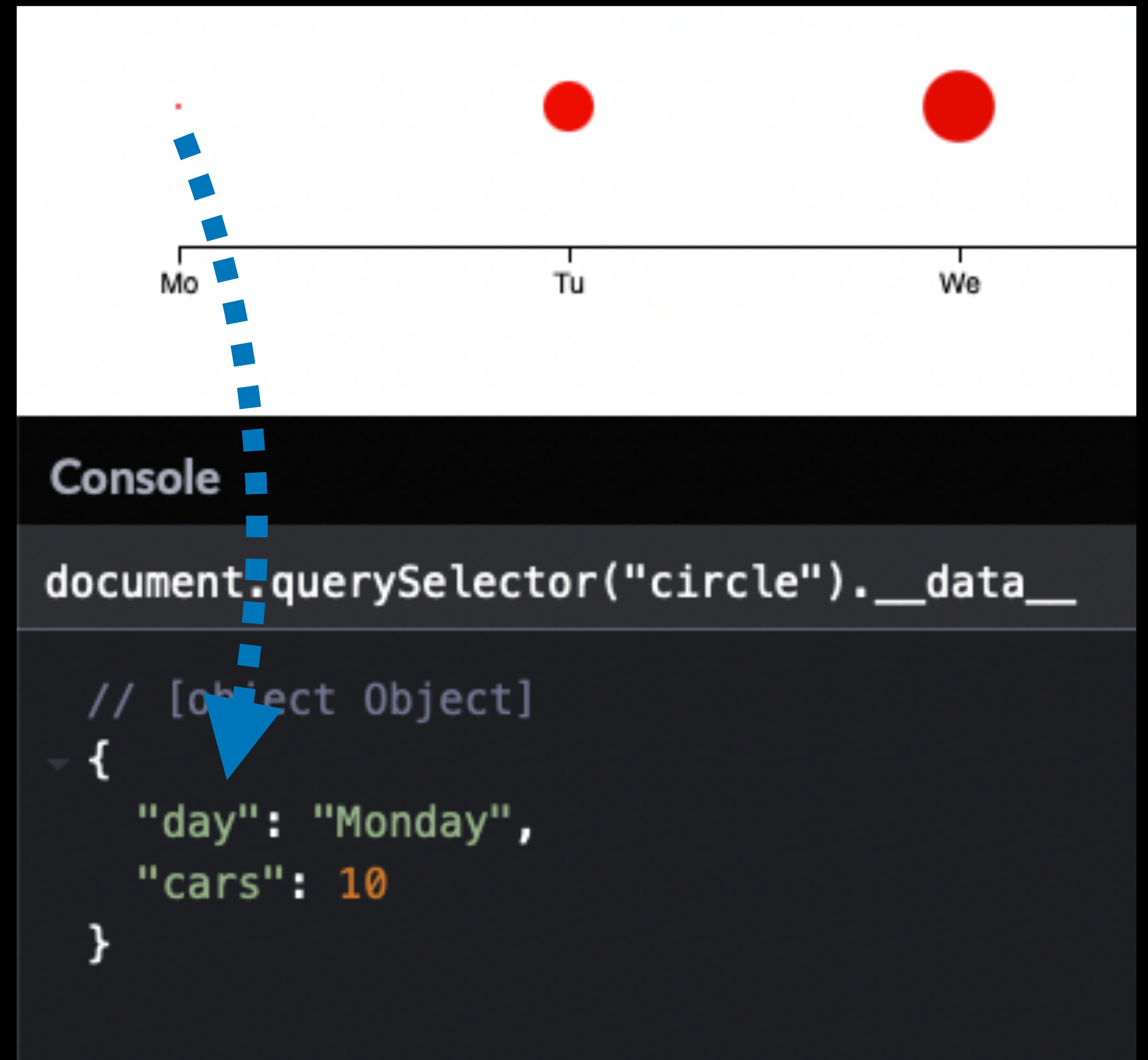
Events

Events help you to add interactivity to your graphs. For example you can add a tooltip or a side panel showing additional details.



Event data

D3 has a magical feature: it adds a `__data__` object to all DOM elements you created so you have access to the original data in your event.



Event binding



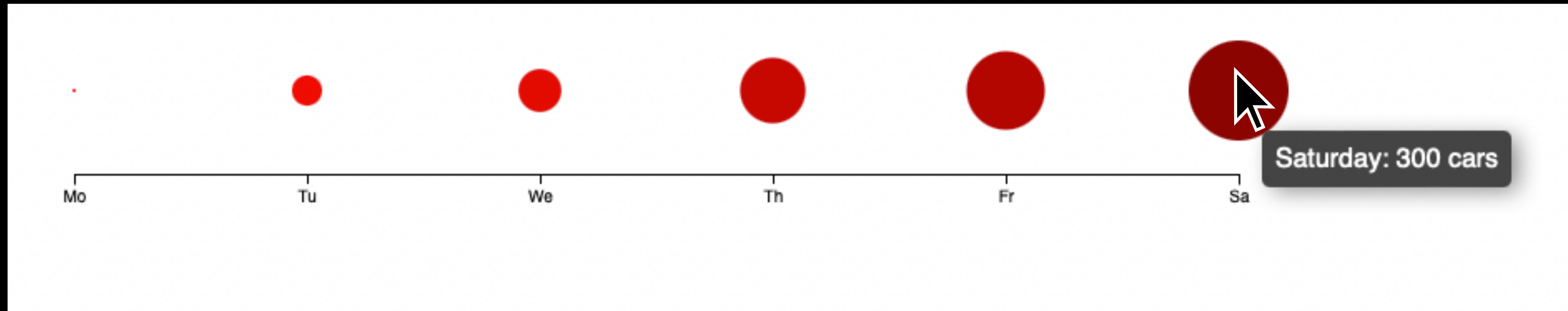
```
d3.select("#scale1")
  .selectAll("circle")
  .data(dataSet)
  .join("circle")
  .on("mouseover", (e, d) =>
    d3.select("#tooltip")
      .style("opacity", 1)
      .text(`${d.day}: ${d.cars} cars`)
  )
  .on("mousemove", (e) =>
    d3
      .select("#tooltip")
      .style("left", e.pageX + 15 + "px")
      .style("top", e.pageY + 15 + "px")
  )
```

You add events by calling `d3.on()`. D3 will call your event function with two parameters:

1. Event data
2. Object data used during `d3.join()`

Tooltip demo

<https://codepen.io/dandevri/pen/azdrEQb>



Huiswerk

- Maak je visualisatie

Schedule

1. Events

2. Modules (import and export)

3. Async data fetching

4. All together!



Package manager

npm is a package manager for the JavaScript programming language. It is the default package manager for [...] Node.js. It consists of a command line client, also called npm, and an online database of [...] packages, called the npm registry.

Neutron Polarization Manipulator

ProductsPricingDocumentationCommunity

npm

Search packages

Search

Sign Up

Sign In

Wondering what's next for npm? [Check out our public roadmap!](#) »

camel-case

TS

4.1.2 • Public • Published 2 months ago

Readme

Explore BETA

2 Dependencies

534 Dependents

20 Versions

Downloads 36M/month

minzipped size 718 B

Transform into a string with the separator denoted by the next word capitalized.

Installation

npm install camel-case --save

Usage

```
import { camelCase } from "camel-case";  
  
camelCase("string"); // => "string"
```

Install

> npm i camel-case

Weekly Downloads

10,314,181

Version	License
4.1.2	MIT
Unpacked Size	Total Files
14.3 kB	15
Issues	Pull Requests
11	0

Homepage

The **npm** website



Import modules

```
import * as d3 from "d3";
```

```
import { onMount } from 'svelte';
```


Modules



```
async function request(url) {  
  let res = await fetch(url);  
  return await res.json();  
}  
  
export default request
```



export default

Modules




```
import CONFIG from './config.js';  
import request from './request.js';  
import makeHtml from './make.js';  
  
const data = await request(CONFIG.url);
```

Import default

Modules

- `require`: Function-based syntax:

javascript

 Copy code


```
const module = require('module-name');
```

CommonJS



- `import`: Declarative syntax (similar to other languages):

javascript

 Copy code

```
import module from 'module-name';
```


ES Modules



Modules

- You can export a function or variable from any file
- There are two types of exports, named and default

Named exports




```
// Individually

export const name = "Robert";
export const age = 29;

// All at once as an object

const name = "Robert";
const age = 29;

export { name, age }
```

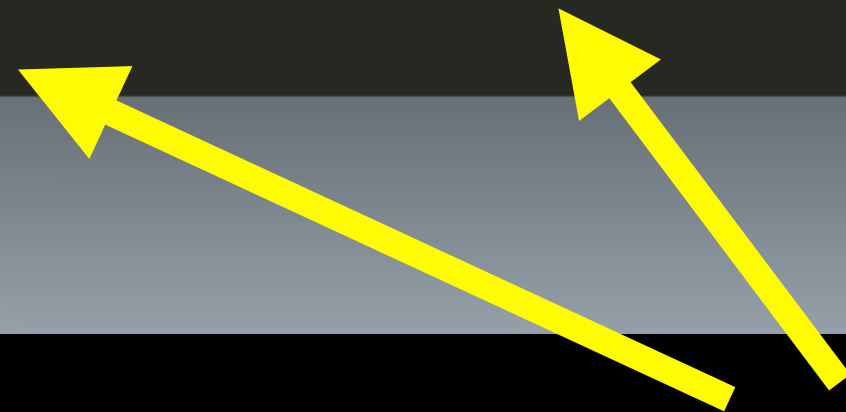


Exporting as an object, due to the {}

Named imports

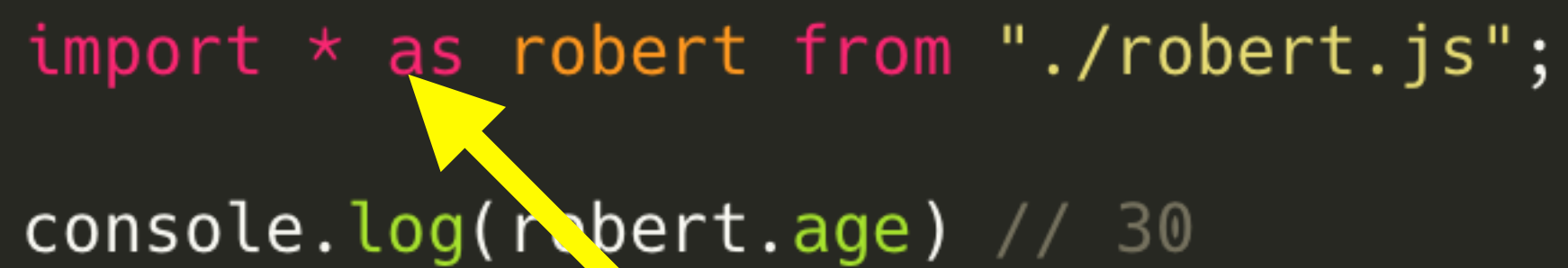


```
import { name, age } from "./robert.js";
```



Importing as an object, due to the {}

Named imports



```
import * as robert from './robert.js';  
console.log(robert.age) // 30
```

Import everything using “as” to name it

Import modules

```
import * as d3 from "d3";
```

```
import { scales, selection } from 'd3';
```

Components

`component === module`

`import Bar from '../components/Bar.svelte';`

Modules = functional programming

Referential transparency: The function always gives the same return value for the same arguments. This means that the function cannot depend on any mutable state

Side-effect free: The function cannot cause any side effects. Side effects may include I/O (e.g., writing to the console or a log file), modifying a mutable object, reassigning a variable, etc.

Schedule

1. Events
2. Modules (import and export)
- 3. Async data fetching**
4. All together!



SYNCHRONOUS LOAD



ASYNCHRONOUS LOAD



Fetching in JavaScript

- ❖ **Callbacks (XMLHttpRequest)**
- ❖ **Promises (Fetch)**
- ❖ **Async / Await (Fetch)**

Sync vs async

why?

Getting data from a resource (API) takes time. It needs to fetch the resource, parse it etc. But also, what if the data isn't available (no internet connection e.g.) how should *errors* be handled?

Separation of concerns

1. Data fetchen (fetch API)
2. Data transformeren (functional, map filter)
3. Renderen van de Charts (d3, scales)

Utility function

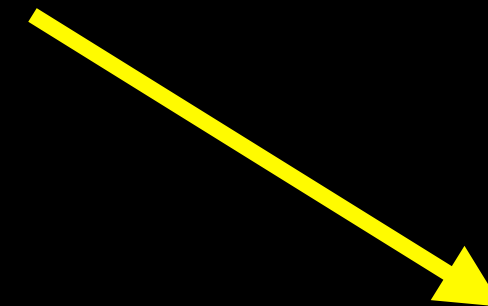
dataFetch.js



dataClean.js



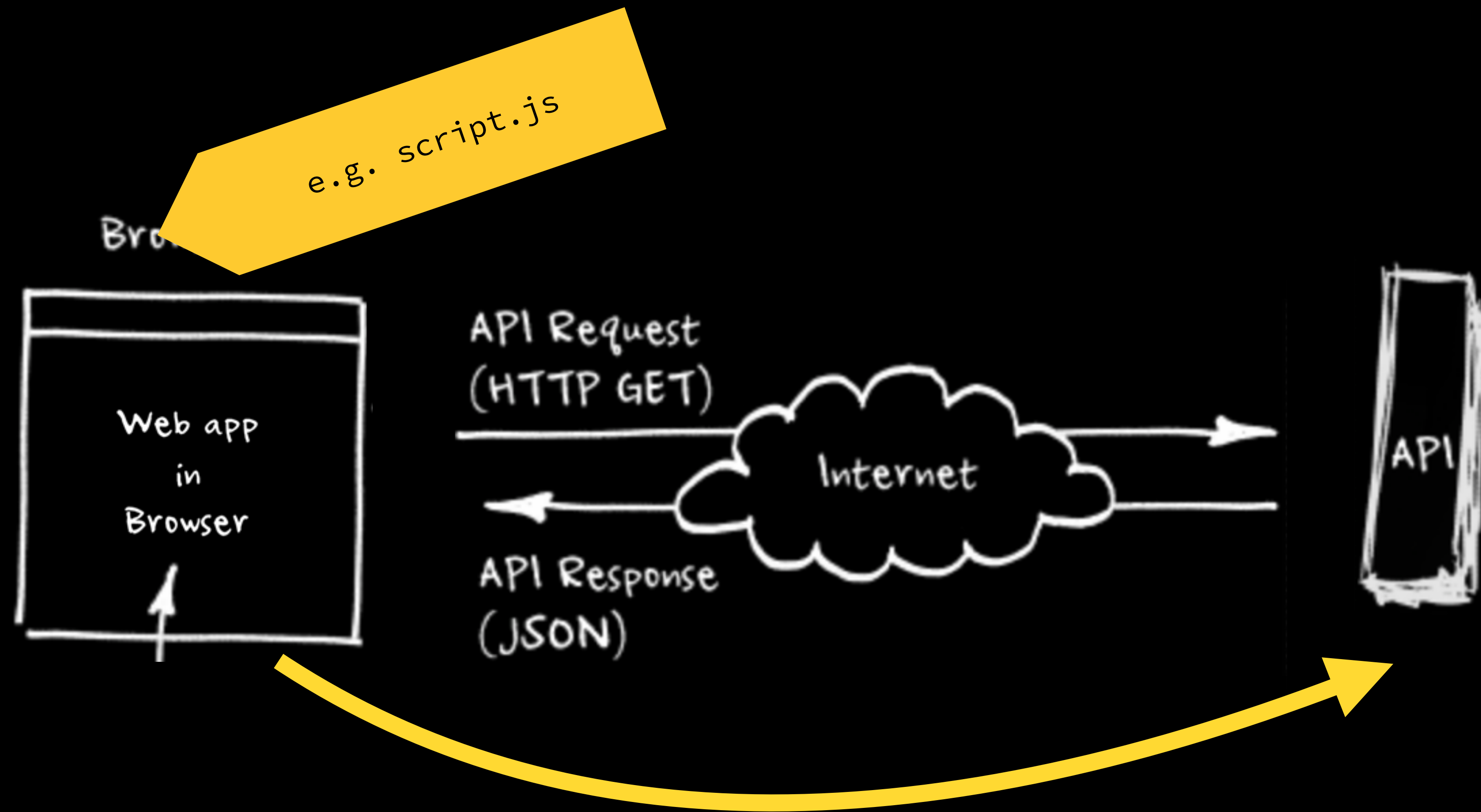
barChart.svelte



pieChart.svelte

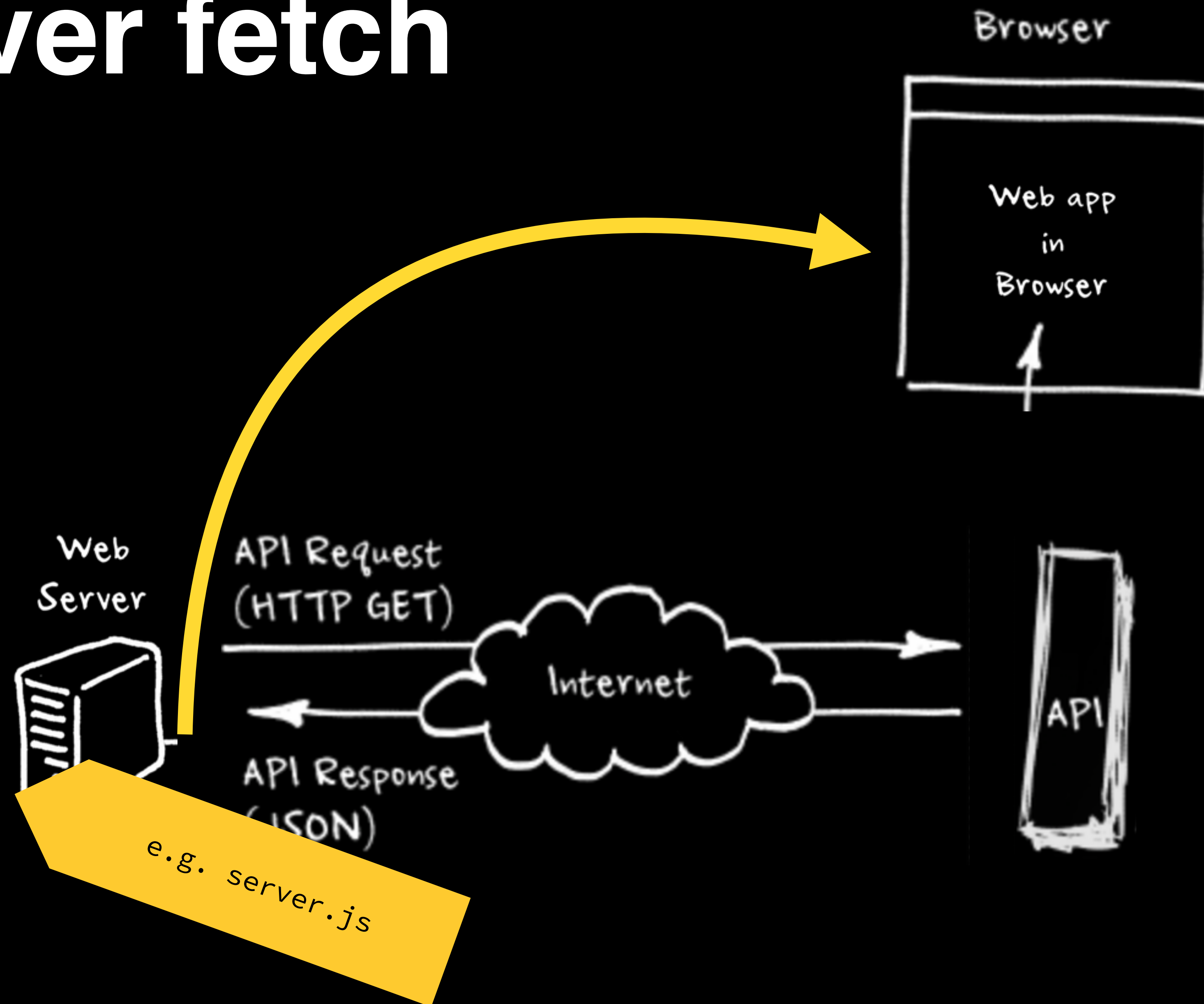
Client fetch

API



Server fetch

API



.env

```
# .env file  
USER_ID="239482"  
USER_KEY="foobar"  
NODE_ENV="development"
```

server.js

```
require('dotenv').config();  
  
process.env.USER_ID; // "239482"  
process.env.USER_KEY; // "foobar"  
process.env.NODE_ENV; // "development"
```

Data fetch module in Svelte

Methode	Reactief	Bereik	Gebruik
1. Stores	✓	Globaal	App-brede state
2. Import/export	✗	Globaal	Constanten, helpers
3. Load + data	✓ (SSR)	Pagina/layout	Serverdata, API's

<https://svelte.dev/docs/kitchen-sink/load>

week-3

dataLoading

.svelte-kit

node_modules

node_modules

static

.gitignore

.npmrc

jsconfig.json

package-lock.json

package.json

README.md

svelte.config.js

vite.config.js

Schedule

1. Events
2. Modules (import and export)
3. Async data fetching
- 4. All together!**



Huiswerk

- * Kies een strategie voor data fetchen
- * Zorg ervoor dat je fetchen async doet
- * Doe functioneel (pure functie met return)

**Uncaught SyntaxError
Unexpected end of input**