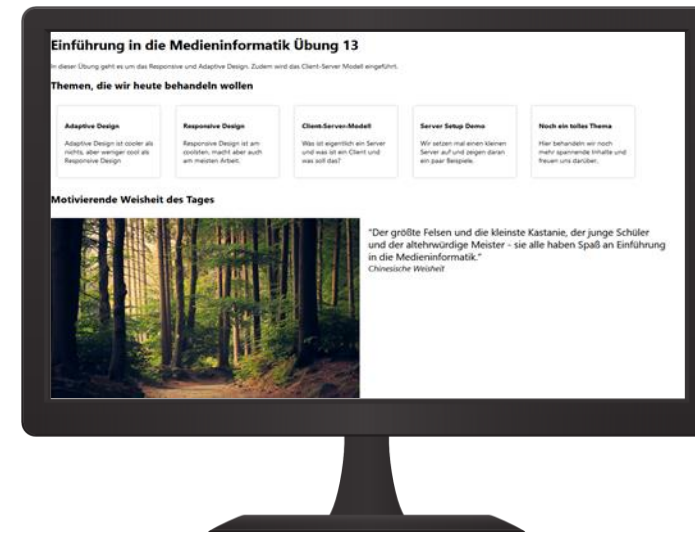
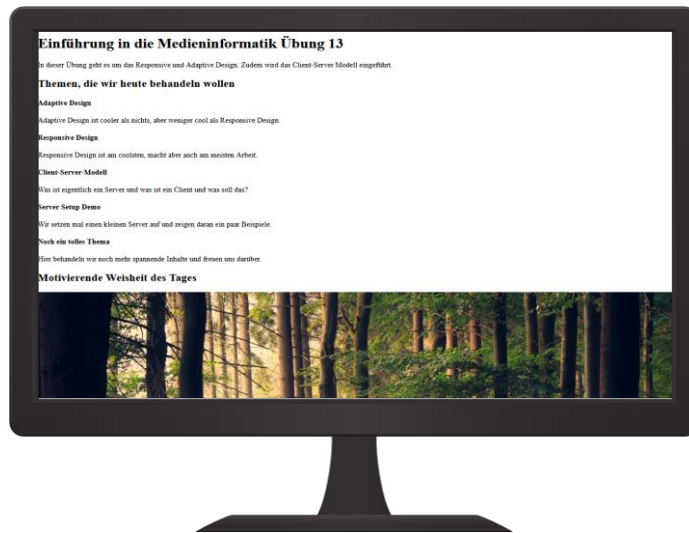


Wir können schon:

Mit HTML die Struktur einer Seite festlegen

Mit CSS das Aussehen und die Positionierung der Elemente verändern.



Wir können schon:

Mit HTML die Struktur einer Seite festlegen

Mit CSS das Aussehen und die Positionierung der Elemente verändern.

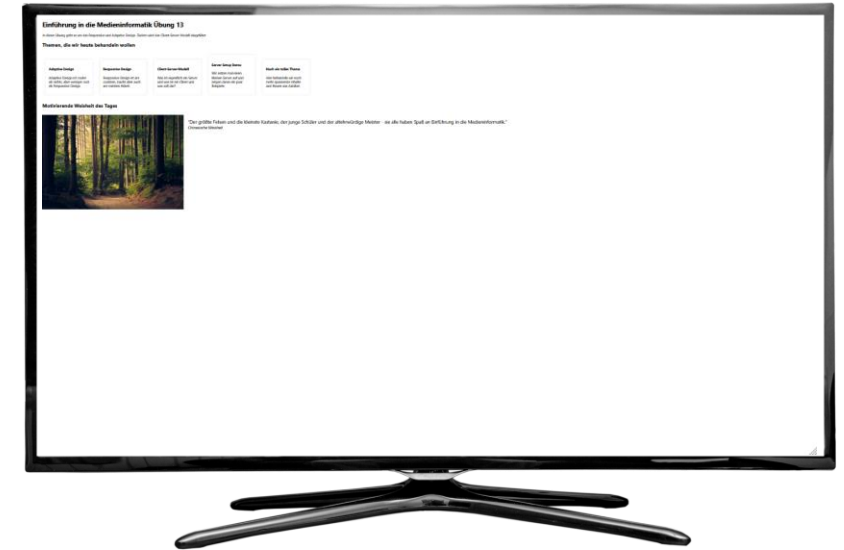
Aber wie gut funktioniert unsere Website auf anderen Geräten?



Modernes Smartphone mit hoher Pixeldichte



Älteres Smartphone mit geringerer Pixeldichte



4K-Fernseher



Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

Unsere Website funktioniert nicht gut auf Geräten mit verschiedenen Größen oder Pixeldichten



## Früher

Fast alle Geräte PCs mit i.d.R. 800x600 Pixeln


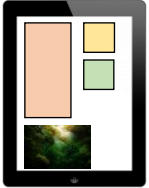
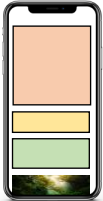
## Heute

Anzeigegeräte sind Smartphones, Laptops, Fernseher, Armbanduhren, Ultra-Wide Monitore, etc.

- Verschiedene Bildschirmgrößen und Seitenverhältnisse
- Sehr große Unterschiede bei der Anzahl an Pixeln
- Browser/Betriebssysteme simulieren manchmal andere Zahl an Pixeln als tatsächlich verfügbar ist

## Die Lösung

Adaptive oder Responsive Design

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
<p>Früher</p> <p>800x600 Pixel</p> <p>↓</p> <p>Heute</p> <p>Viele verschiedene Geräte Unterschiedliche Größen Unterschiedliche #Pixel</p> <p>↓</p> <p>Lösung</p> <p>Adaptive oder Responsive Design</p>	<p><b>Adaptive Design - Ansatz</b></p> <p>Definieren von Layouts für verschiedene Displaygrößen</p> <p>z.B. ein Layout für Desktops, eines für Tablets und eines für Smartphones</p> <p>Wechsel zwischen den Ansichten bei festgelegten Displaygrößen in Pixeln</p> <div><p><b>Vorteile</b></p><ul style="list-style-type: none"><li>+ Es müssen nur wenige, feste Designs entworfen werden</li><li>+ Technisch unkompliziert</li><li>+ Zeitsparende Umsetzung</li></ul></div> <div><p><b>Nachteile</b></p><ul style="list-style-type: none"><li>- Nur für bestimmte Geräte/Viewports optimiert</li><li>- Häufig mehr CSS-Code als notwendig</li><li>- Häufige Fehldarstellungen auf abweichenden Endgeräten</li></ul></div>		 <p>Layout für Displays mit Breite &gt; 1200px</p>  <p>Layout für Displays mit Breite zwischen 600px und 1200px</p>  <p>Layout für Displays mit Breite &lt; 600px</p>

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
<p>Früher 800x600 Pixel</p> <p>↓</p> <p>Heute Viele verschiedene Geräte Unterschiedliche Größen Unterschiedliche #Pixel</p> <p>↓</p> <p>Lösung Adaptive oder Responsive Design</p>	<h2>Responsive Design - Ansatz</h2> <p>Kombination von Adaptive Design mit flüssigen Gestaltungsrastern</p> <p>Layout wird so optimiert, dass der zur Verfügung stehende Raum optimal ausgenutzt wird</p> <div><h3>Vorteile</h3><ul style="list-style-type: none"><li>+ Jede Displaygröße wird berücksichtigt</li><li>+ Zukünftige Geräte werden automatisch mit abgedeckt</li><li>+ Es wird kein Platz verschenkt</li></ul></div> <div><h3>Nachteile</h3><ul style="list-style-type: none"><li>- Komplexere technische Umsetzung</li><li>- Mockups, Wireframes und Skizzen können das flüssige Verhalten schwer darstellen</li><li>- Komplexeres Design</li></ul></div>		

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Media Queries

Erlauben Definition von CSS-Regeln nur für bestimmte Endgeräte.

style.css

```
@media (max-width: 600px) {  
  /* Hier könnte Ihr css stehen */  
}
```

**Schlüsselwort**

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Media Queries

Erlauben Definition von CSS-Regeln nur für bestimmte Endgeräte.

style.css

```
@media (max-width: 600px) {  
    /* Hier könnte Ihr css stehen */  
}
```

Bedingung

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Media Queries

Erlauben Definition von CSS-Regeln nur für bestimmte Endgeräte.

Mögliche Bedingungen sind Art des Geräts, Höhe oder Breite des Viewports u.v.m.

style.css

```
@media (max-width: 600px) {  
    /* Styles für Mobiltelefone */  
}  
  
@media only print {  
    /* Styles für Ausdrucken */  
}
```



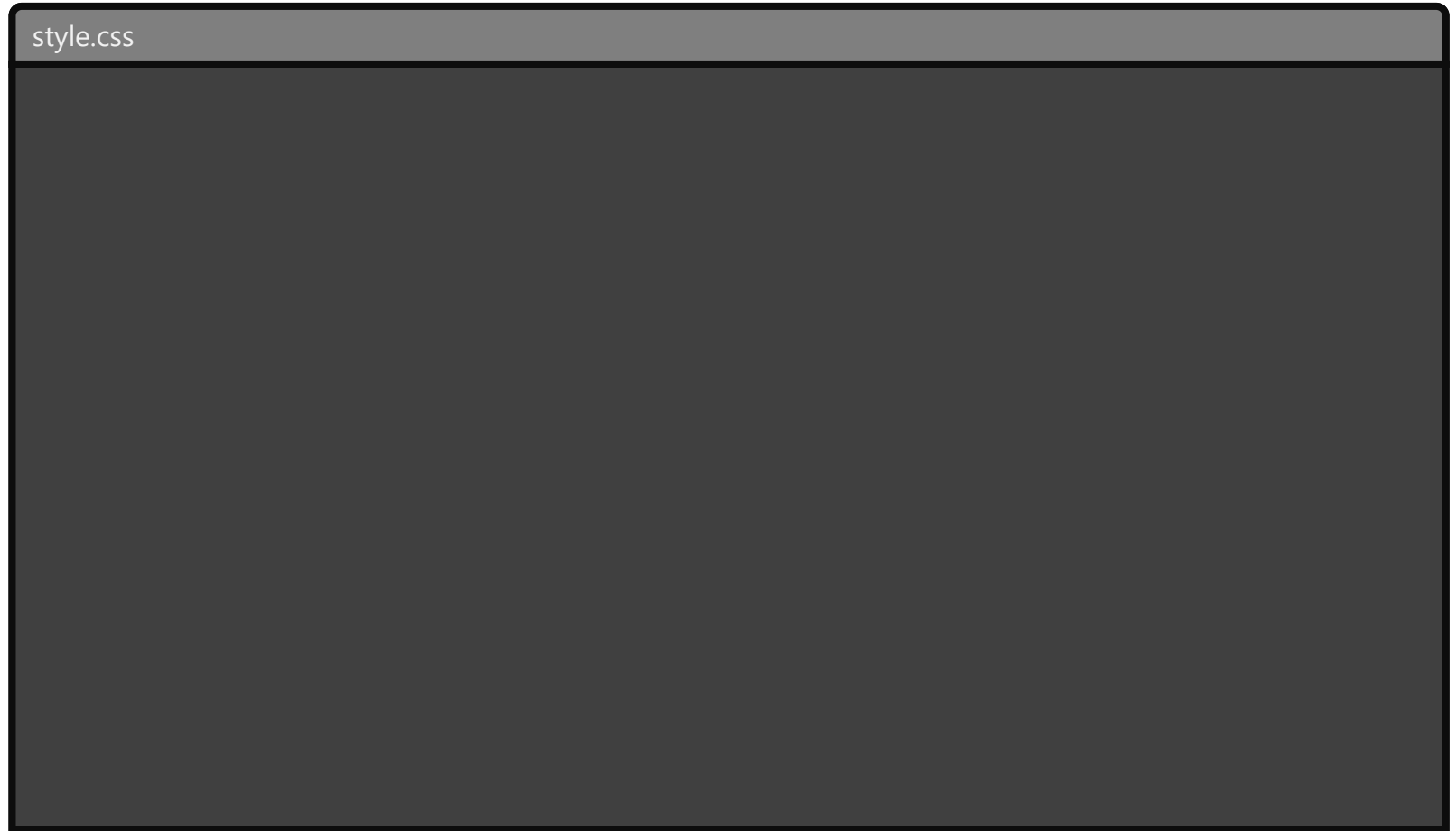
Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Media Queries

Erlauben Definition von CSS-Regeln nur für bestimmte Endgeräte.

Mögliche Bedingungen sind Art des Geräts, Höhe oder Breite des Viewports u.v.m.

Können logisch kombiniert werden.



Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Spaltenlayout

Häufig werden Oberflächen mit Spalten realisiert

Oft verwendet: 12 Spalten

CSS-Klassen bestimmen, wie viele Spalten ein Objekt einnehmen darf

style.css

```
col-1 { width: 8.33%; }  
col-2 { width: 16.66%; }  
col-3 { width: 25%; }  
col-4 { width: 33.33%; }  
col-5 { width: 41.66%; }  
col-6 { width: 50%; }  
col-7 { width: 58.33%; }  
col-8 { width: 66.66%; }  
col-9 { width: 75%; }  
col-10 { width: 83.33%; }  
col-11 { width: 91.66%; }  
col-12 { width: 100%; }
```

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Umsetzung: Spaltenlayout

Häufig werden Oberflächen mit Spalten realisiert

Oft verwendet: 12 Spalten

CSS-Klassen bestimmen, wie viele Spalten ein Objekt einnehmen darf

style.css

```
col-1 { width: 8.33%; }  
col-2 { width: 16.66%; }  
col-3 { width: 25%; }  
col-4 { width: 33.33%; }  
col-5 { width: 41.66%; }  
col-6 { width: 50%; }  
col-7 { width: 58.33%; }  
col-8 { width: 66.66%; }  
col-9 { width: 75%; }  
col-10 { width: 83.33%; }  
col-11 { width: 91.66%; }  
col-12 { width: 100%; }
```

```
<body>  
  <div class=„red col-12“></div>  
  <div class=„blue col-6“></div>  
  <div class=„green col-6“></div>  
</body>
```



Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Idee für adaptives Spaltenlayout

spalten.css

```
@media (max-width: 600px) {  
  col-s-1 { width: 8.33%; }  
  ...  
  col-s-6 { width: 50%; }  
  ...  
  col-s-12 { width: 100%; }  
}  
  
@media (max-width: 1200px) {  
  col-m-1 { width: 8.33%; }  
  ...  
  col-m-12 { width: 100%; }  
}  
  
Media (min-width: 1200px) {  
  col-1 { width: 8.33%; }  
  ...  
  col-12 { width: 100%; }  
}
```

spalten.html

```
<div class=„red col-12“>  
</div>  
<div class=„blue col-6 col-s-12“>  
</div>  
<div class=„green col-6 col-s-12“>  
</div>
```

**Default:** 6 Spalten (halbe Breite)  
**Kleines Display:** Ganze Breite

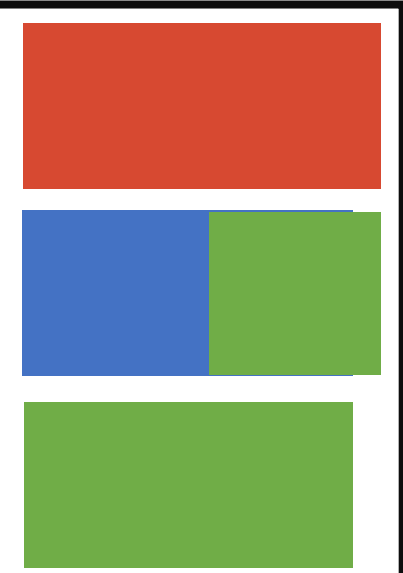
600px

1200px

Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

## Idee für adaptives Spaltenlayout

spalten.css	spalten.html
<pre>@media (max-width: 600px) {   col-s-1 { width: 8.33%; }   ...   col-s-6 { width: 50%; }   ...   col-s-12 { width: 100%; } }  @media (max-width: 1200px) {   col-m-1 { width: 8.33%; }   ...   col-m-12 { width: 100%; } }  Media (min-width: 1200px) {   col-1 { width: 8.33%; }   ...   col-12 { width: 100%; } }</pre>	<pre>&lt;div class=„red col-12“&gt; &lt;/div&gt; &lt;div class=„blue col-6 col-s-12“&gt; &lt;/div&gt; &lt;div class=„green col-6 col-s-12“&gt; &lt;/div&gt;</pre> <div><b>Default:</b> 6 Spalten (halbe Breite) <b>Kleines Display:</b> Ganze Breite</div>



Problemstellung	Adaptive Design	Responsive Design	Umsetzung
-----------------	-----------------	-------------------	-----------

Weitere Konzepte -> fortgeschritten, aber sehr mächtig

## Flexbox (`display:flex`)

Container mit dieser CSS-Eigenschaft passen ihre Kinder automatisch dem verfügbaren Platz an

Aktuell eines der wichtigsten Responsive-Design Tools

Parameter wie wrapping, Abstände, Größen der einzelnen Kinder, etc. können zusätzlich konfiguriert werden

Ausführlicher Guide dazu mit guten Erklärungen:  
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

## CSS Grid (`display:grid`)

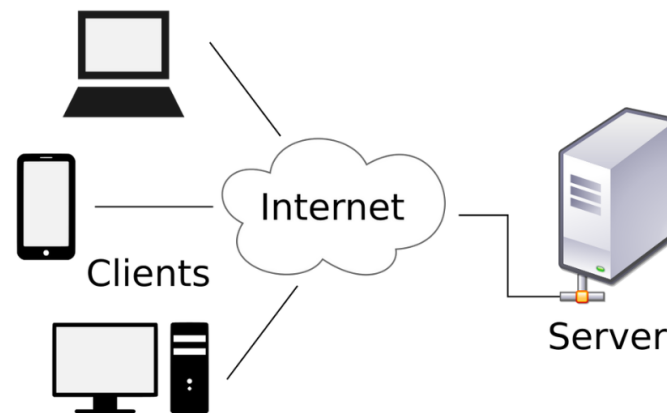
Erlaubt, Kinder des Containers in einem Grid anzuordnen, dessen Größe sich dynamisch anpasst

Einzelne Zeilen und Spalten des Grids können in der Größe relativ zueinander angepasst werden

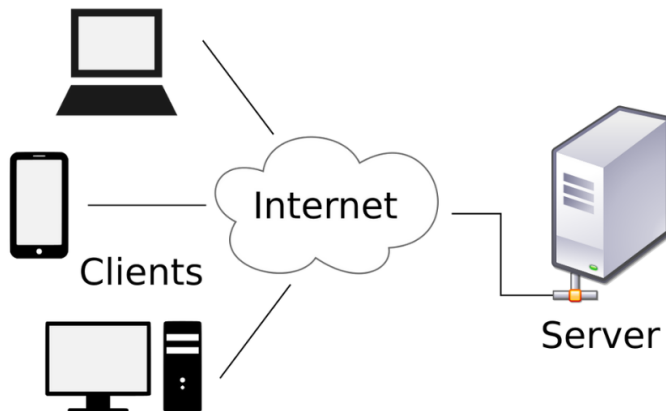
Kinder können mehrere Zeilen/Spalten füllen

Ausführlicher Guide dazu mit guten Erklärungen:  
<https://css-tricks.com/snippets/css/complete-guide-grid/>

Häufig vermitteltes Bild



Häufig vermitteltes Bild



Server sind zum Beispiel...

- ... Gameserver
- ... der Rechner auf dem ISIS läuft
- ... Server von Zoom/Discord/etc.
- ...

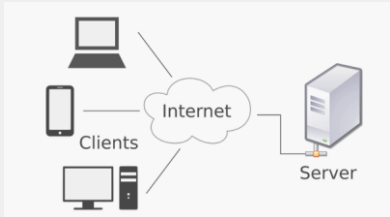
Clients sind zum Beispiel

- ... Smartphones
- ... Laptops
- ... PCs
- ...

Und alles ist *irgendwie* per Internet verbunden



### Häufig vermitteltes Bild



Server sind zum Beispiel...

- ... Gameserver
- ... der Rechner auf dem ISIS läuft
- ... Googles Rechenzentren
- ...

Clients sind zum Beispiel...

- ... Smartphones
- ... Laptops
- ... PCs
- ...

Und alles ist *irgendwie*  
per Internet verbunden

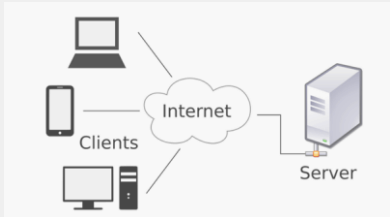
### Andere Beispiele für Server:

- Euer Router
- Visual Studio Code mit WSL
- Jupyter Notebooks

### Andere Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint

### Häufig vermitteltes Bild



Server sind zum Beispiel...

- ... Gameserver
- ... der Rechner auf dem ISIS läuft
- ... Googles Rechenzentren
- ...

Clients sind zum Beispiel...

- ... Smartphones
- ... Laptops
- ... PCs
- ...

Und alles ist *irgendwie*  
per Internet verbunden

### Andere Beispiele für Server:

- Euer Router
- Visual Studio Code mit WSL
- Jupyter Notebooks

### Andere Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint

## Umgangssprachlich

**Server** = Rechner von anderen Leuten oder Institutionen, die Dienste bereitstellen.

**Clients** = die Geräte, mit denen wir diese Dienste benutzen.

## Tatsächlich

**Server** = Jedes Programm, das eine Netzanfrage beantwortet

**Clients** = Jedes Programm, das eine Netzanfrage sendet

Server & Client können sich im gleichen Netzwerk oder auf dem gleichen Rechner befinden (2 verschiedene Programme). Ein Programm kann sogar beides sein.

**Server** = Jedes Programm,  
das eine Netzanfrage  
beantwortet

**Clients** = Jedes Programm,  
das eine Netzanfrage sendet

### Beispiele für Server:

- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles  
Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn

**Server** = Jedes Programm,  
das eine Netzanfrage  
beantwortet

**Clients** = Jedes Programm,  
das eine Netzanfrage sendet

### Beispiele für Server:

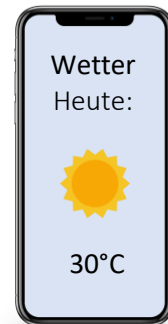
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



**Server** = Jedes Programm,  
das eine Netzanfrage  
beantwortet

**Clients** = Jedes Programm,  
das eine Netzanfrage sendet

### Beispiele für Server:

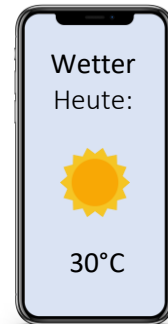
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

**Beispiele für Server:**

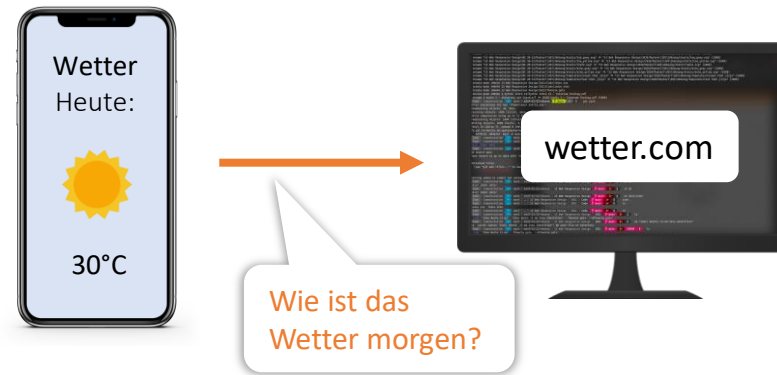
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

**Beispiele für Clients:**

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



**Server** = Jedes Programm,  
das eine Netzanfrage  
beantwortet

**Clients** = Jedes Programm,  
das eine Netzanfrage sendet

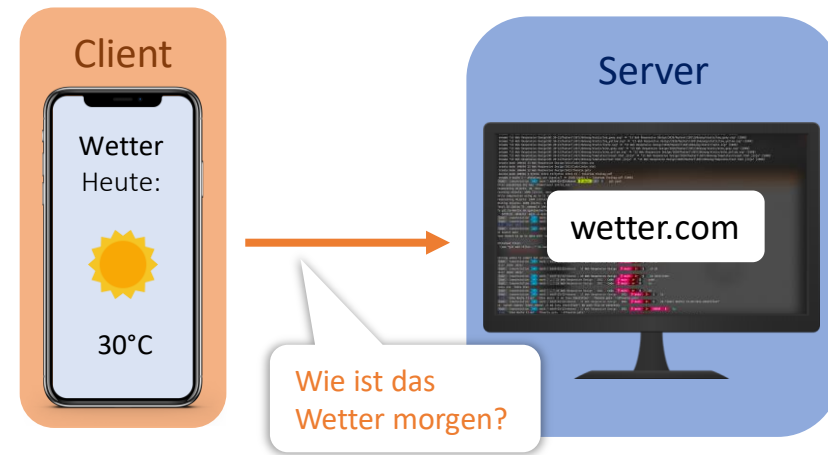
### Beispiele für Server:

- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*  
-> Diese Aussage macht genau genommen keinen Sinn



**Server** = Jedes Programm,  
das eine Netzanfrage  
beantwortet

**Clients** = Jedes Programm,  
das eine Netzanfrage sendet

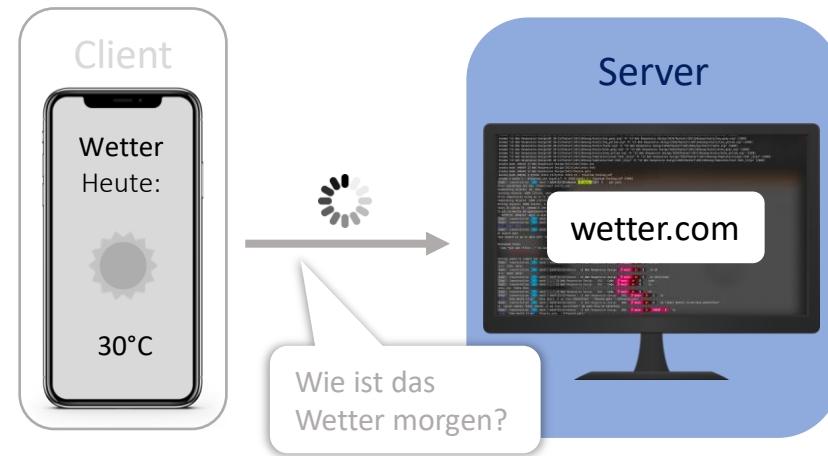
### Beispiele für Server:

- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*  
-> Diese Aussage macht genau genommen keinen Sinn





Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

#### Beispiele für Server:

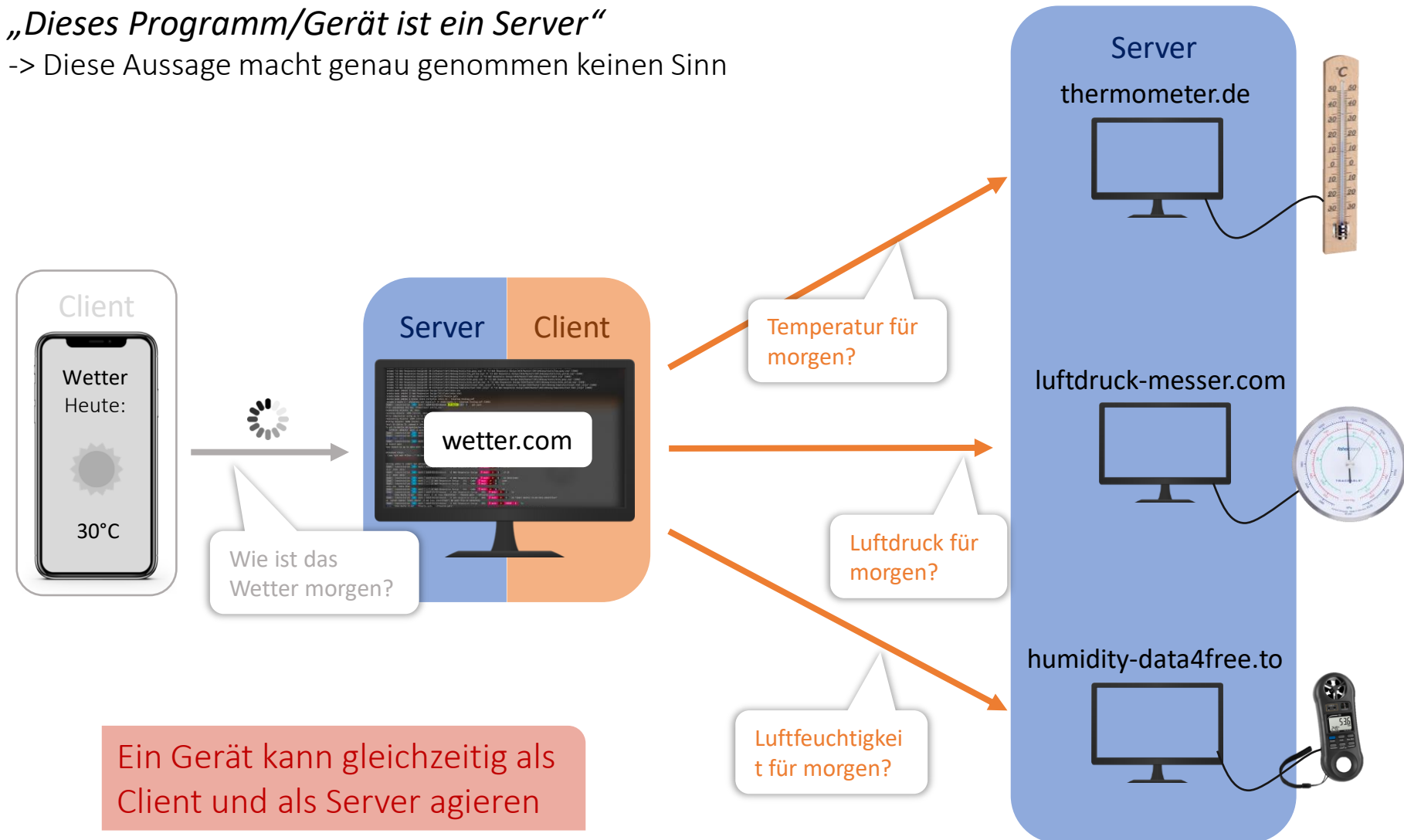
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

#### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

#### Beispiele für Server:

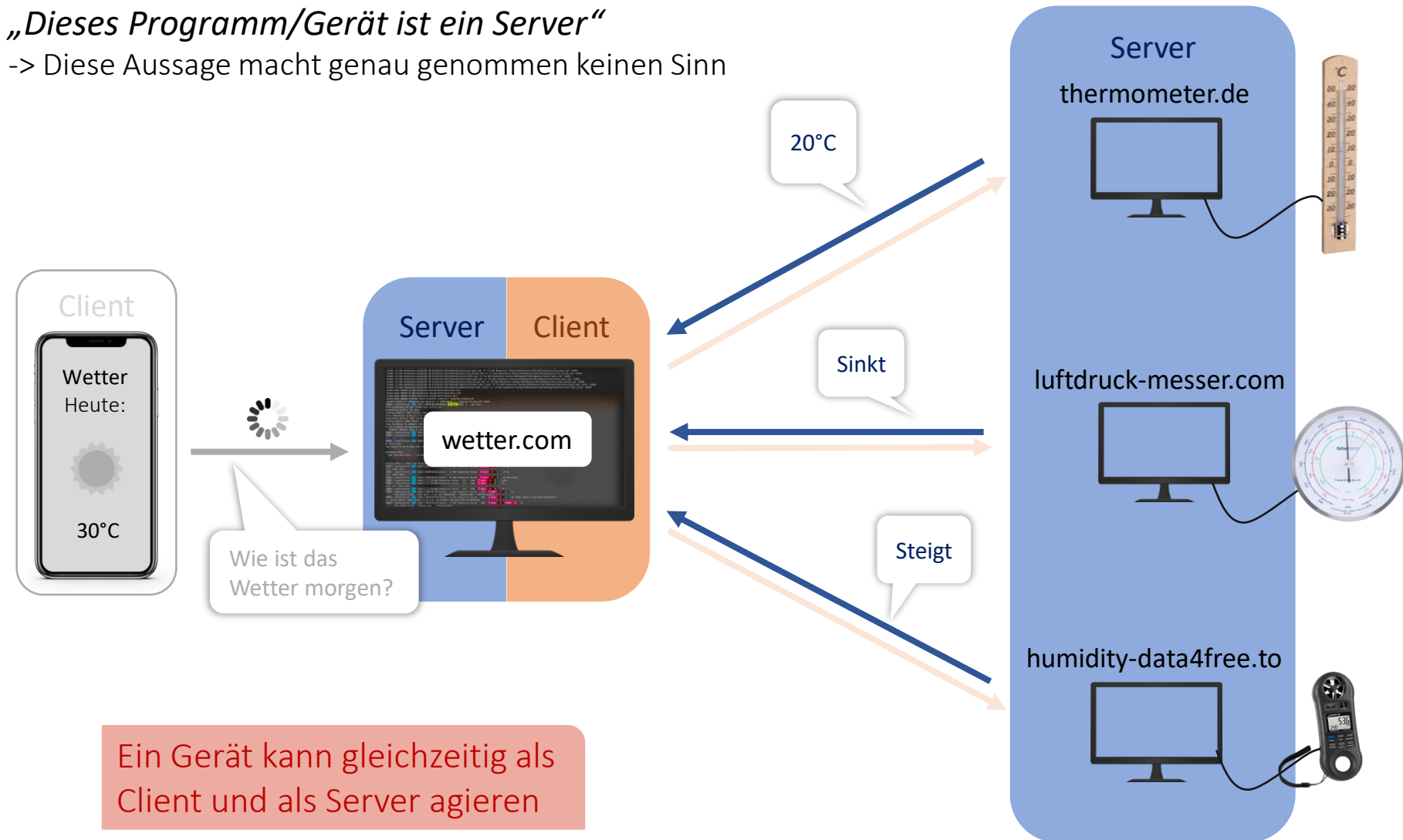
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

#### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

**Beispiele für Server:**

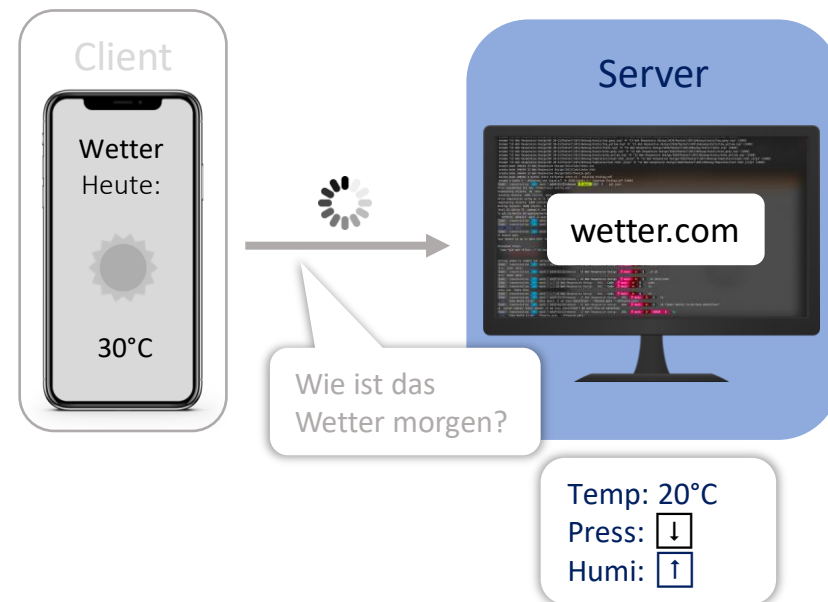
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

**Beispiele für Clients:**

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Ein Gerät kann gleichzeitig als  
Client und als Server agieren



Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

**Beispiele für Server:**

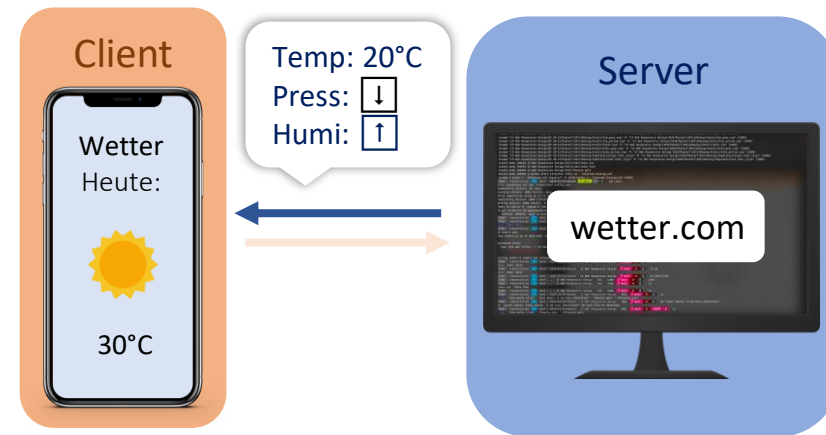
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

**Beispiele für Clients:**

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Ein Gerät kann gleichzeitig als  
Client und als Server agieren



**Server** = Jedes Programm, das eine Netzanfrage beantwortet

**Clients** = Jedes Programm, das eine Netzanfrage sendet

### Beispiele für Server:

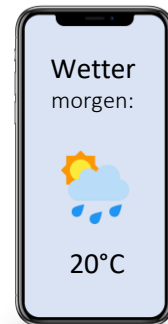
- Euer Router
- VS Code mit WSL
- Jupyter Notebooks
- ISIS-Server
- Discord/Zoom/etc.
- Gameserver

### Beispiele für Clients:

- Googles Rechenzentren
- Jupyter Notebooks
- PowerPoint
- Smartphones
- Laptops
- PCs

*„Dieses Programm/Gerät ist ein Server“*

-> Diese Aussage macht genau genommen keinen Sinn



Ein Gerät kann gleichzeitig als Client und als Server agieren



luftdruck-messer.com



humidity-data4free.to

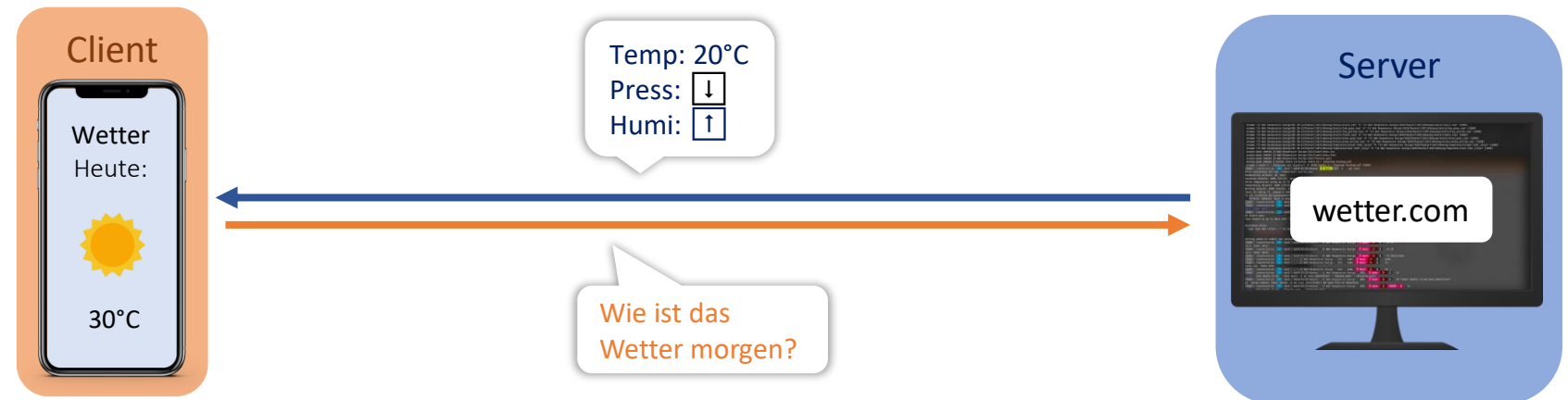


Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

Ein Gerät kann gleichzeitig  
als Client und als Server  
agieren.

## Zurück zum Beispiel von gerade eben

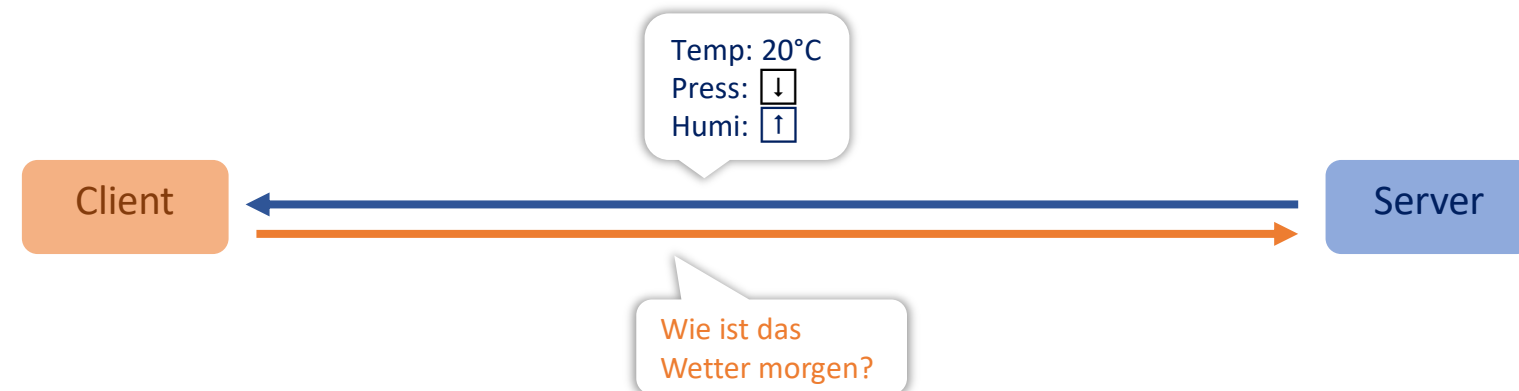


Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

Ein Gerät kann gleichzeitig  
als Client und als Server  
agieren.

Zurück zum Beispiel von gerade eben  
Was wird da eigentlich übertragen?



Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

Ein Gerät kann gleichzeitig  
als Client und als Server  
agieren.

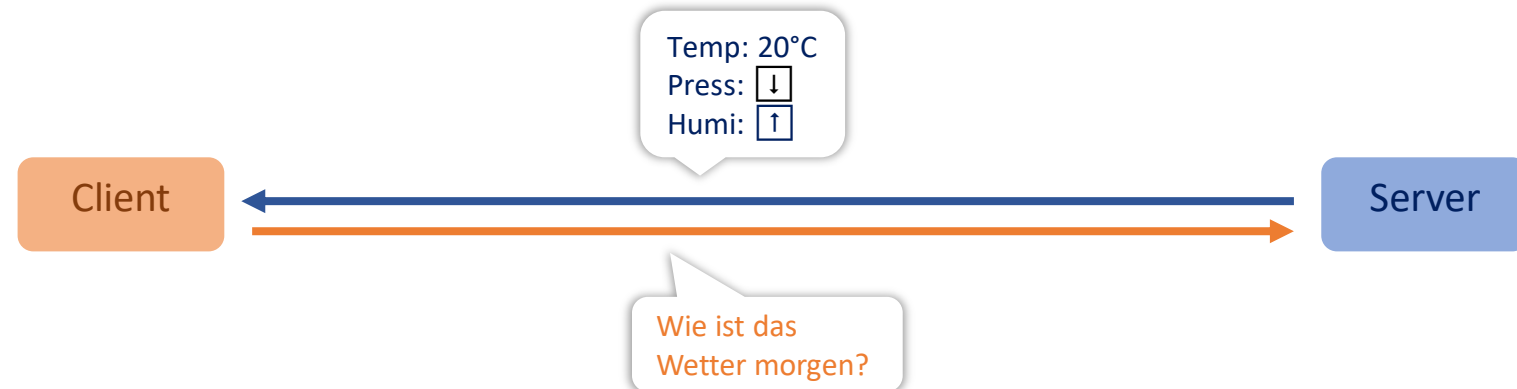
## Zurück zum Beispiel von gerade eben

Was wird da eigentlich übertragen?

Man kann jede Art von Daten übertragen -> Wichtig ist nur, dass der Empfänger sie lesen kann

Die meiste Kommunikation im Web benutzt das **HTTP-Protokoll** (einziges, das wir betrachten)

**HTTP überträgt nur einfachen Text**





Server = Jedes Programm,  
das eine Netzanfrage  
beantwortet

Clients = Jedes Programm,  
das eine Netzanfrage sendet

Ein Gerät kann gleichzeitig  
als Client und als Server  
agieren.

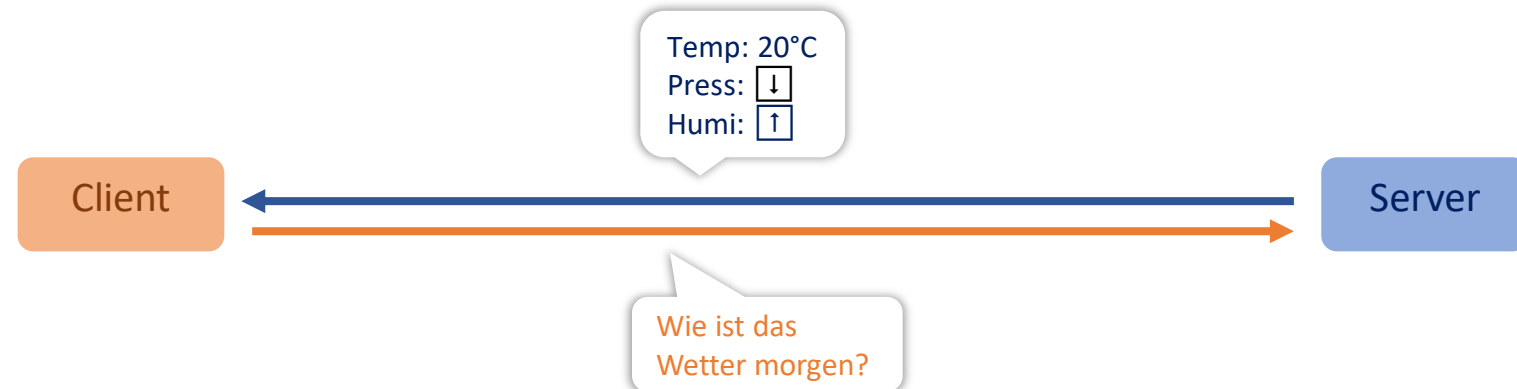
## Zurück zum Beispiel von gerade eben

Was wird da eigentlich übertragen?

Man kann jede Art von Daten übertragen -> Wichtig ist nur, dass der Empfänger sie lesen kann

Die meiste Kommunikation im Web benutzt das **HTTP-Protokoll** (einziges, das wir betrachten)

**HTTP überträgt nur einfachen Text**



Man kann jede Art von Daten übertragen

-> Wichtig ist nur, dass der Empfänger sie lesen kann

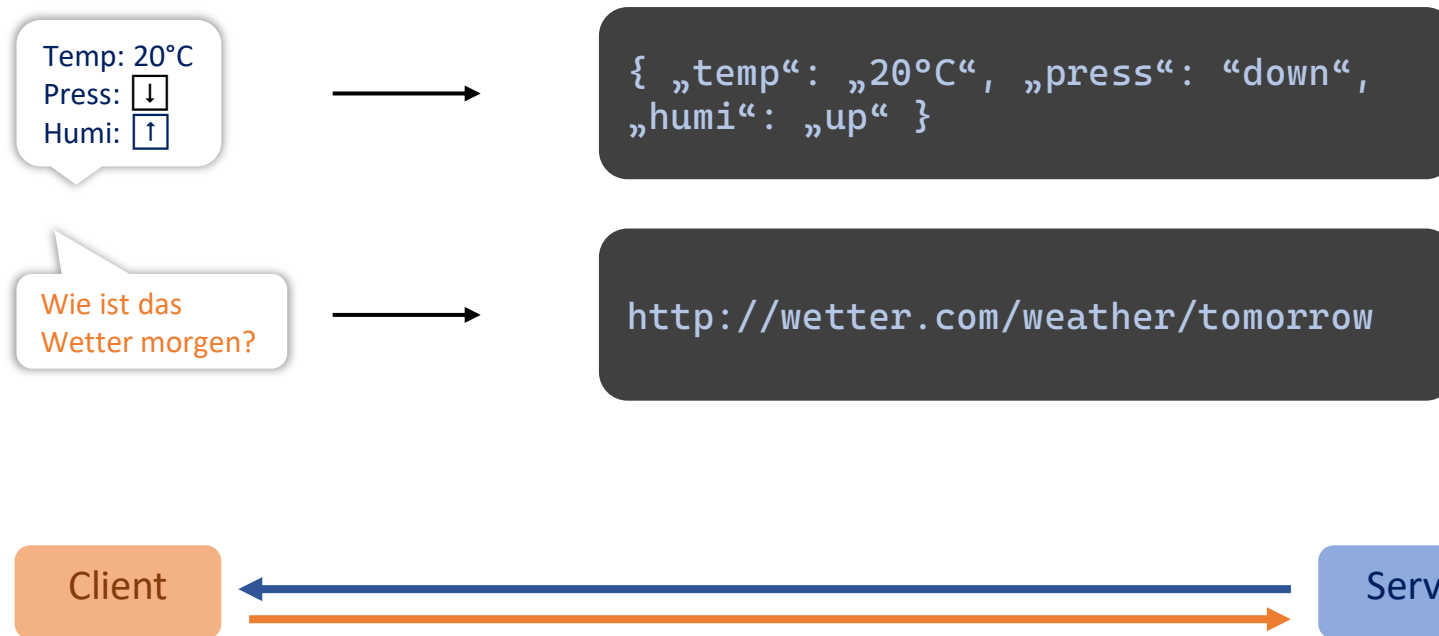
Die meiste Kommunikation im Web benutzt das HTTP-Protokoll (einziges, das wir betrachten)

HTTP überträgt nur einfachen Text

## Zurück zum Beispiel von gerade eben

Was wird da eigentlich übertragen?

Anfragen werden als **URLs** gestellt. Daten werden in maschinenlesbares **JSON** übersetzt.



Man kann jede Art von Daten übertragen

-> Wichtig ist nur, dass der Empfänger sie lesen kann

Die meiste Kommunikation im Web benutzt das HTTP-Protokoll (einziges, das wir betrachten)

HTTP überträgt nur einfachen Text

Anfragen werden als URLs gestellt. Daten werden in maschinenlesbares JSON übersetzt.

### Zurück zum Beispiel von gerade eben

Was wird da eigentlich übertragen?

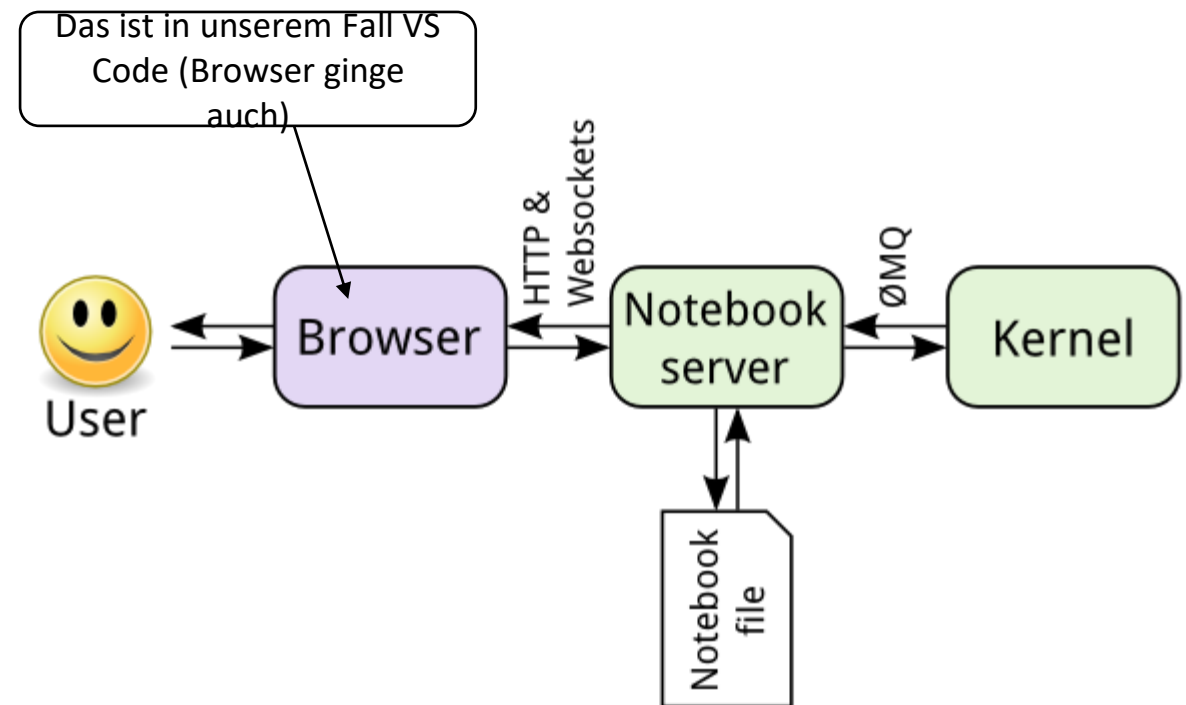
Der **Client** muss jetzt diese JSON-Daten darstellen können (z.B. als HTML-Elemente).

Der **Server** muss die URL richtig lesen und beantworten können.



## Beispiel: Jupyter Notebooks

1. Ein geeignetes Programm (z.B. VS Code) öffnet eine .ipynb-Datei
2. Das Programm startet den *Notebook Server*, der .ipynb-Dateien interpretieren kann
3. Das Programm agiert als Client und stellt eine Oberfläche bereit, in die wir python-Code eingeben können
4. Wenn wir eine Zelle abschicken, wird der Code an den *Notebook Server* gesendet. Dieser benutzt den Python-Kernel um ihn auszuführen
5. Das Ergebnis schickt der *Notebook Server* als Text an den Client (VS Code) zurück -> er wird nutzerfreundlich angezeigt



## Rückblick - und was jetzt?

Wir haben Websites mit **HTML** aufgebaut und mit **CSS** gestaltet.

So können wir Medien und Informationen visualisieren und aufbereiten.

Wir haben das **Client-Server-Modell** kennengelernt und einen kleinen Einblick in die Funktionsweise des Internets bekommen.

Damit können wir unsere Kreationen Anderen zeigen – auf der ganzen Welt!

Wir kennen jetzt die zentralen Konzepte, die das Internet zum größten und interaktivsten Wissensspeicher aller Zeiten machen!

## Rückblick - und was jetzt?

Wir haben Websites mit **HTML** aufgebaut und mit **CSS** gestaltet.

So können wir Medien und Informationen visualisieren und aufbereiten.

Wir haben das **Client-Server-Modell** kennengelernt und einen kleinen Einblick in die Funktionsweise des Internets bekommen.

Damit können wir unsere Kreationen Anderen zeigen – auf der ganzen Welt.

Wir kennen jetzt die zentralen Konzepte, die das Internet zum größten und interaktivsten Wissensspeicher aller Zeiten machen!

Einige Fragen sind noch offen

Wie programmiere ich einen Server?

Wie bringe ich ihn „ins Internet“?

Wie schreibe ich eine komplexe Webanwendung?

Muss man im Internet nicht auch noch auf Sicherheit achten?

Woher weiß mein Gerät, wohin es seine Anfrage senden soll, wenn ich Google aufrufe?

Und 1000 Weitere...

## Rückblick - und was jetzt?

Einige Fragen sind noch offen

Wie programmiere ich einen Server?

Wie bringe ich ihn „ins Internet“?

Wie schreibe ich eine komplexe  
Webanwendung?

Muss man im Internet nicht auch noch auf  
Sicherheit achten?

Woher weiß mein Gerät, wohin es seine  
Anfrage senden soll, wenn ich Google aufrufe?

Und 1000 Weitere...

## Rückblick - und was jetzt?

Einige Fragen sind noch offen



Wie programmiere ich einen Server?

Wie bringe ich ihn „ins Internet“?

Wie schreibe ich eine komplexe Webanwendung?

Muss man im Internet nicht auch noch auf Sicherheit achten?

Woher weiß mein Gerät, wohin es seine Anfrage senden soll, wenn ich Google aufrufe?

Und 1000 Weitere...

Webtechnologien im 3. Semester

(oder wann auch immer ihr es machen wollt 😊)

Ideen für Selbststudium:

JavaScript Einführung von w3schools.com

<https://www.w3schools.com/js/default.asp>

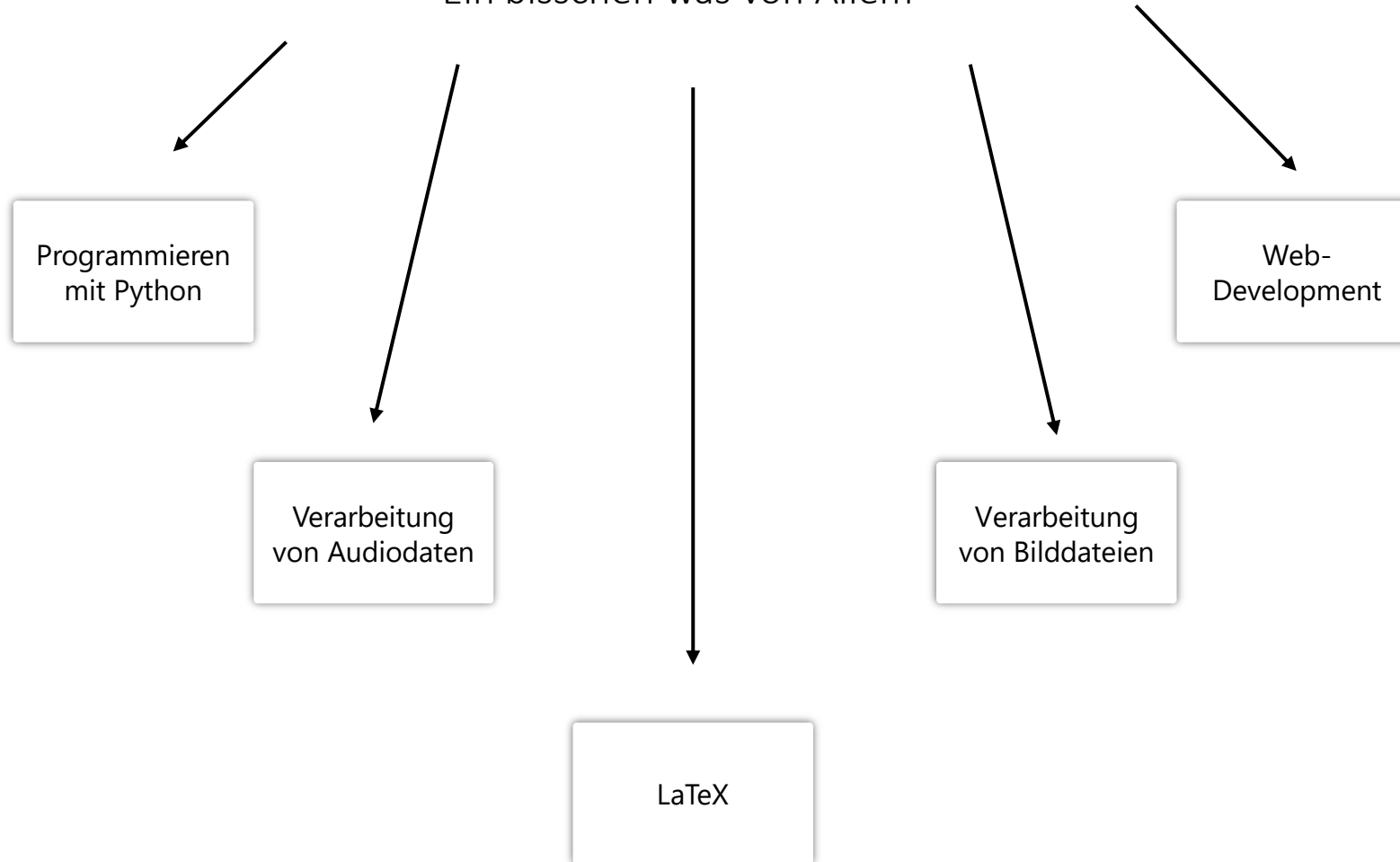
Baut euch eine Website und ladet sie bei einem kostenfreien Hostinganbieter hoch. Zum Beispiel:

- <https://spaces.w3schools.com/> (Account erforderlich)
- <https://pages.github.com/> (Account erforderlich, Grundkenntnisse in Git sehr hilfreich)
- [TU Berlin persönliche Websites](#) (bei eurem TUB-Account inklusive, Einrichtung nicht ganz trivial)



## Rückblick - und was jetzt?

Das war *Einführung in die Medieninformatik*  
Ein bisschen was von Allem



## Rückblick - und was jetzt?

Das war *Einführung in die Medieninformatik*  
Ein bisschen was von Allem

Programmieren  
mit Python

Verarbeitung  
von Audiodaten

LaTeX

Verarbeitung  
von Bilddateien

Web-  
Development

Die Grundlagen kennt ihr.

Studium heißt jetzt:  
macht das weiter, was euch am besten gefallen hat!