HTML5-APIs

4 ½ Technologien kurz vorgestellt

Im Fokus

1. Drag & Drop
2. File API
3. Canvas 2D
4. Webcam-Feeds (User Media + HTML5-Video)
5. **Versammeltes Vorwissen?**

1. Drag & Drop

Satans eigene API, direkt aus der Internet Explorer-Hölle

„The drag-and-drop API is horrible [...]“

- Ian Hickson, Editor der HTML5-Specs

„[...] a steaming pile of bovine manure.“

Peter-Paul Koch, Browsermacken-Feinschmecker

Die API wurde zu 80% aus dem **Internet Explorer 5** in den Standard übernommen!

Was kann die API?

1. Drag & Drop innerhalb eines Browserfensters
2. Drag & Drop zwischen mehreren Browserfenstern
3. Drag & Drop aus dem System in ein Browserfenster

Einfache Theorie

```
<!-- Beispiel -->
<div id="Ziel">
  Hier hin!
</div>

<script>
  var ziel = document.querySelector('#Ziel');
  ziel.ondrop = function(evt){
    alert(evt.dataTransfer);
  };
</script>
```

Standardverhalten: Redirect

- Browser interpretieren Datei-Drops standardmäßig als Redirect-Anweisung
- Gegenmittel: **Abbruch des drop-Events!**

```
ziel.ondrop = function(evt){
  alert(evt.dataTransfer.files);
  evt.preventDefault();
};
```

So funktioniert‘s wirklich

- Grundproblem: **Das Drop-Event findet gar nicht statt!**
- Es gibt Drag-Events, die ihre Folge-Events unterbinden
- Nur wenn die *Stopper-Events abgebrochen werden*, läuft die Operation wie erwartet

```
// Stopper-Event abbrechen
ziel.ondragover = function(evt){
  evt.preventDefault();
};
```

Zusammenfassung

1. Auf dem Ziel-Element `dragover`-Event abfangen und abbrechen
2. Auf dem Ziel-Element `drop`-Event abfangen und …
    1. Das Event abbrechen
    2. Die `dataTransfer`-Eigenschaft des Event-Objekts auslesen

Extra: Dragenter & Dragleave

```
// Farbliche Markierung hinzufügen
ziel.ondragenter = function(evt){
  this.classList.add('active');
};

// Farbliche Markierung entfernen
ziel.ondragleave = function(evt){
  this.classList.remove('active');
};
```

Alles soweit klar?

dataTransfer-Objekt

- Eigenschaft des Event-Objekts von Drag- und Drop-Events
- Meist schreib/lesegeschützt, viele kryptische Properties
- Für uns interessant: **die `files`-Property bei Drop-Events**

```
ziel.ondrop = function(evt){
  console.log(evt.dataTransfer.files);
  evt.preventDefault();
};
```

dataTransfer.files

- **`dataTransfer.files` enthält eine FileList mit den gezogenen Dateien!**
- FileList = Array-ähnliches Objekt mit File-Objekten (repräsentiert die gezogenen Dateien)
- Und das führt uns zum nächsten Thema …

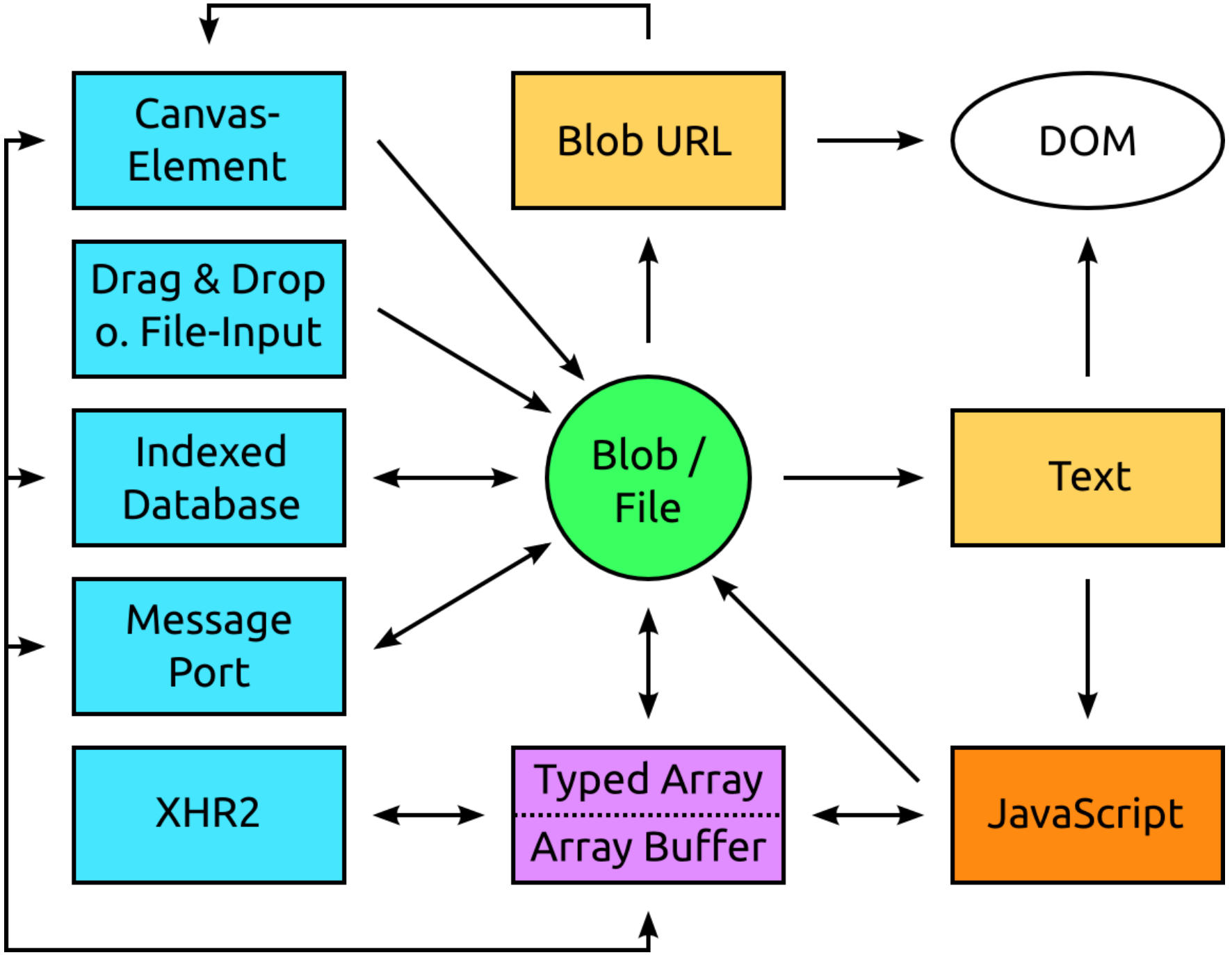
Alles klar zu Drag & Drop?

2. File API

Dateiverarbeitung im Browser

Neu in HTML5

- Dateiezugriff via Drag & Drop oder Input-Element
- Dateien ver/bearbeiten, neue Binärdatentypen (uninteressant)
- **Austausch zwischen vielen verschiedenen HTML5-APIs**


Canvas-
Element
Drag & Drop
o. File-Input
Indexed
Database
Message
Port
XHR2
Blob URL
Blob /
File
Typed Array
Array Buffer
DOM
Text
JavaScript

Dateizugriff

- Variante 1: Input-Feld (`files`-`Property`, uninteressant)
- Variante 2: Drag & Drop
- Ergebnis: **FileList-Objekte**

FileList-Objekte

- Die `files`-Eigenschaften von File-Inputs und Drop-Events enthält ein **FileList**-Objekt
- **FileList**-Objekt = Array-Ähnliche Objekte, die die File-Objekte beinhalten
- Eigenschaften:
  - `length` - Anzahl der Dateien in der Liste
  - `FileList.item(x)` - Getter-Funktion (`fileList[x]` funktioniert auch)

File-Objekte

- Repräsentiert Dateien
- `file.name` - Dateiname
- `file.size` - Länge in Bytes
- `file.type` - MIME-Type
- `file.lastModifiedDate` - Letztes Änderungsdatum der Datei
- `file.slice(start, end [, type])` - Erstellt aus Teilen des Objekts ein Blob-Objekt

Dateien einlesen

FileReader-Objekte lesen Dateien ein

```
var meinReader = new FileReader();
```

FileReader-Methoden

- `readAsText(datei [, charset])`
- `readAsDataURL(datei)`
- `readAsArrayBuffer(datei)`
- `abort()`

```
ziel.ondrop = function(evt){
  var file = evt.dataTransfer.files.item(0);
  var meinReader = new FileReader();
  meinReader.readAsText(file);
  evt.preventDefault();
};
```

FileReader-Einleseprozess

1. Das FileReader-Objekt beginnt das im Aufruf der Einlesemethoden übergebene Blob- oder File-Objekt einzulesen
2. Die Status-Attribute `readyState` und `error` des FileReader-Objekts verändern sich im Laufe der Operation und entsprechende Events feuern
3. Sobald die Operation angeschlossen ist, wird der eingelesene Inhalt der Datei in der `result`-Eigenschaft des FileReader-Objekts zur Verfügung gestellt

FileReader-Events

- `loadstart` - Einleseprozesses startet
- `progress` - Byte eingelesen oder 50 Millisekunden ohne Aktion verstrichen
- `abort` - Einleseprozesses abgebrochen
- `error` - Fehler tritt auf (übergibt Fehler-Objekt)
- `load` - Einleseprozess erfolgreich abgeschlossen
- `loadend` - Einleseprozess beendet, entweder durch Erfolg oder wenn ein Fehler auftrit

```
ziel.ondrop = function(evt){
  var file = evt.dataTransfer.files.item(0);
  var meinReader = new FileReader();
  meinReader.readAsText(file);
  // Feuert wenn fertig eingelesen
  meinReader.onload = function(){
    alert(this.result);
  };
  evt.preventDefault();
};
```

Alles klar zur File API?

3. Canvas 2D

Programmierbare <img>-Elemente

Was wird geboten?

- **Komplette 2D-Zeichen-API!**
- Formen zeichnen, Pfade malen, Animationen …
- Für uns interessant: **Pixel-Manipulation**

Canvas-Basics

```
<canvas id="Foo" width="480" height="280">
  Ersatzinhalt
</canvas>

<script>

  // Canvas-Element
  var canvas = document.getElementById('Foo');

  // 2D-Zeichen-API
  var context = canvas.getContext('2d');

</script>
```

Canvas und Context

- Zu unterscheiden sind **Canvas** und **Context**
- Canvas = DOM-Element mit bemalter Oberfläche
- Context = Zeichen-API (z.B. 2D)

Zeichenoperationen

```
// Style festlegen, Rechteck zeichnen
context.fillStyle = 'rgb(255, 0, 0)';
context.fillRect(20, 40, 240, 160);
// Anderes Rechteck in anderem Style
context.fillStyle = 'blue';
context.fillRect(100, 20, 200, 160);
// Teil wieder löschen
context.clearRect(40, 20, 160, 120);
```

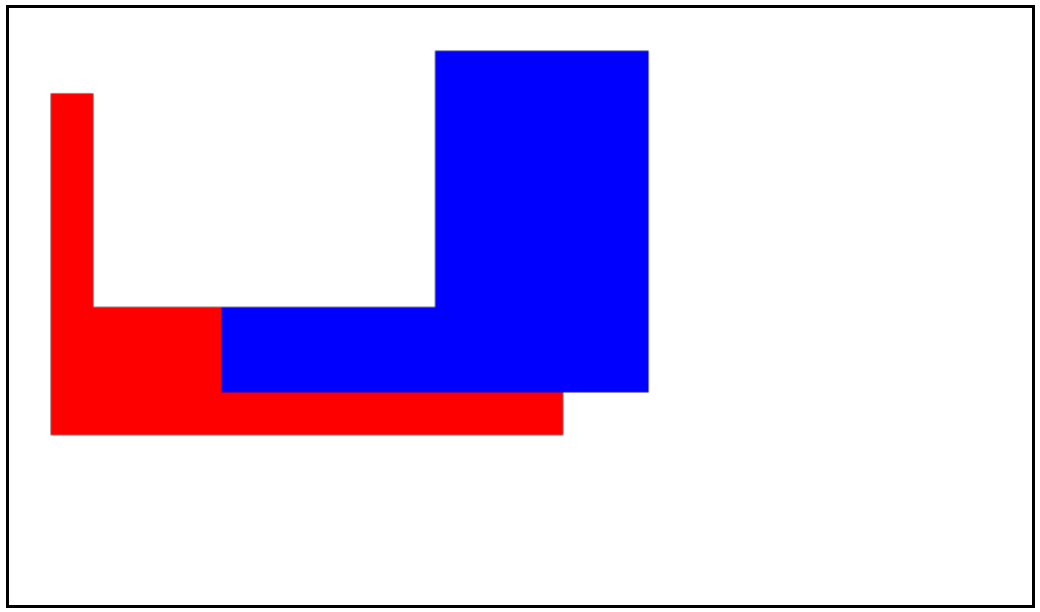


Bild exportieren

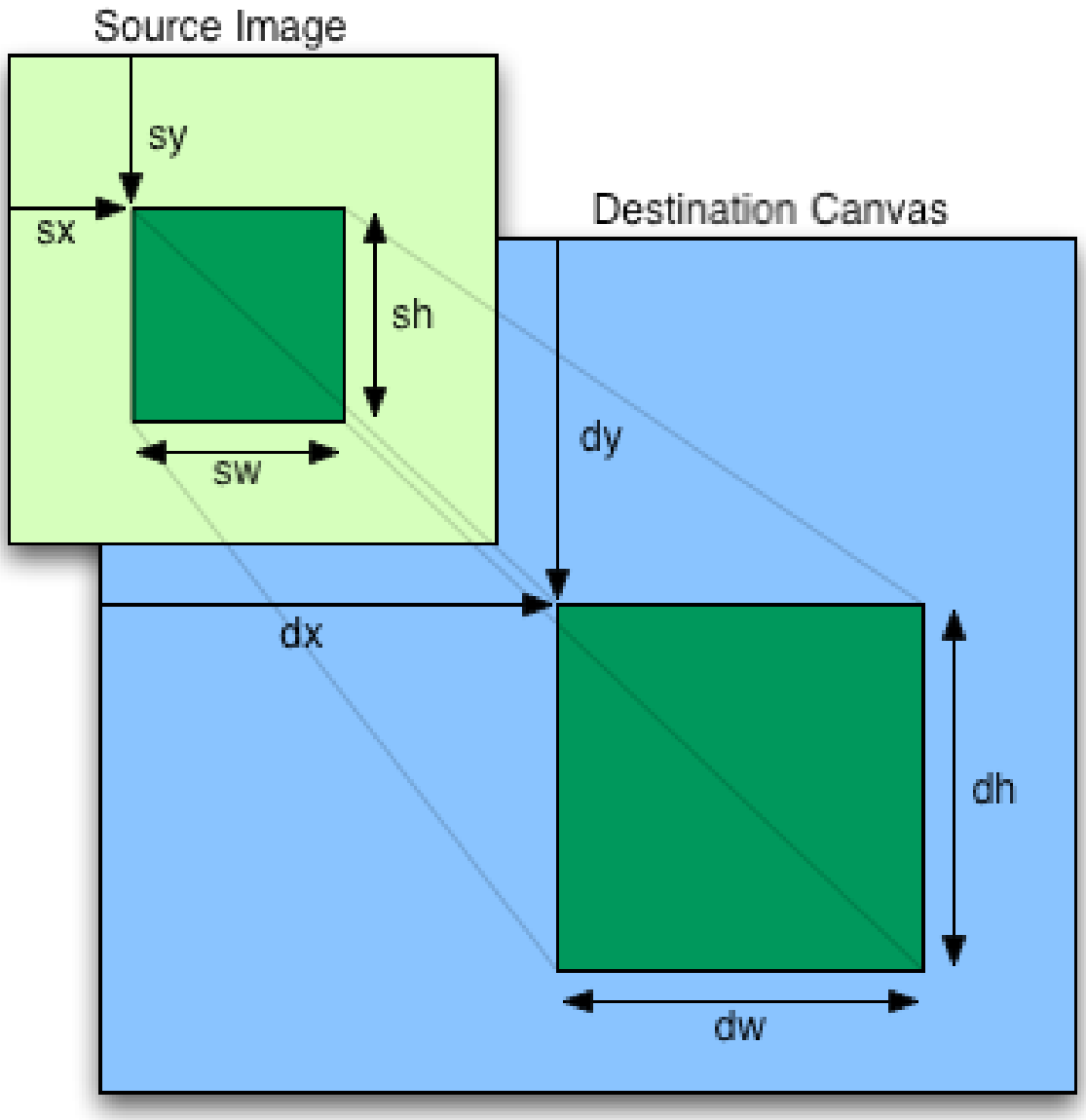
```
// Beachten: Canvas, nicht Context
canvas.toDataURL(); // Data-URL-String
```

Optional: Angabe des Bild-Typs (z.B. "`image/png`")

Bilder auf die Canvas zeichnen

Drei Möglichkeiten:

1. `context.drawImage(img, dx, dy)`
2. `context.drawImage(img, dx, dy, dw, dh)`
3. `context.drawImage(img, sx, sy, sw, sh, dx, dy, dw, dh)`

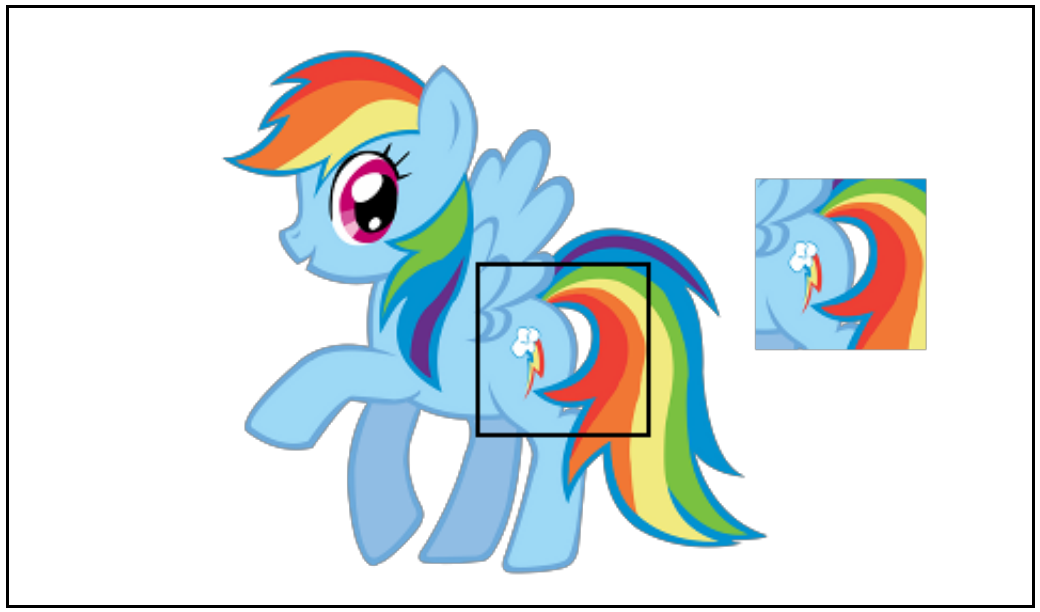

Source Image
sy
sx
sh
sw
Destination Canvas
dy
dx
dh
dw

```
// Bild-Objekt erstellen
var img = new Image();
img.src = "test.png";

// Erst zeichnen wenn das Bild geladen ist
img.onload = function(){
  context.drawImage(img, 100, 20);
};
```

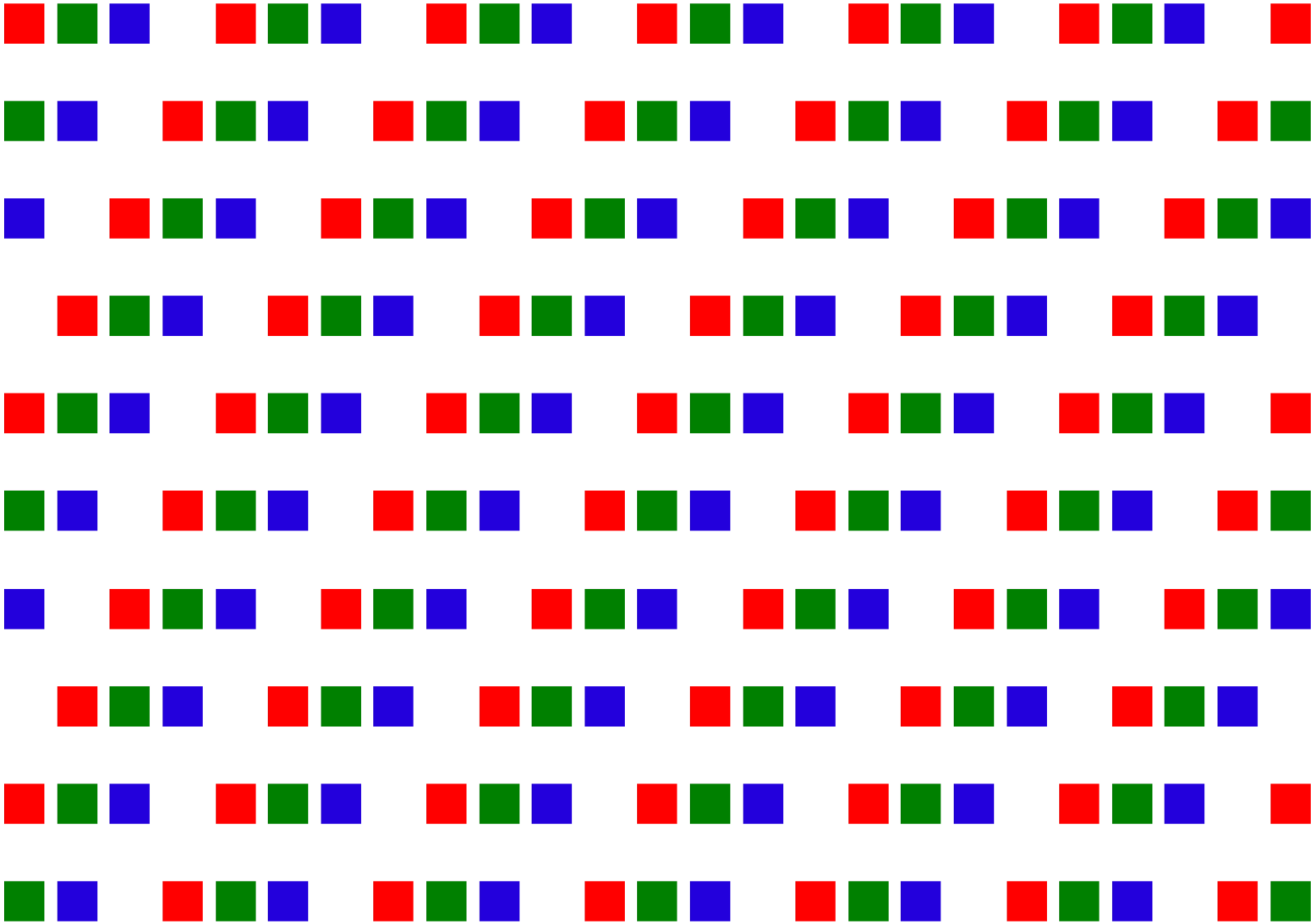
Canvas-Ausschnitte kopieren und einfügen

- Kopieren: `getImageData(x, y, w, h)`
- Rückgabewert: `imageData`-Objekt mit den Bildinformationen des Ausschnitts
- Einfügen: `putImageData(imageData, x, y)`
- Selbst bauen: `createImageData(x, y)`



Das imageData-Objekt

- Eigenschaften `width`, `height`
- **Die Eigenschaft `data` enthält die Abfolge der Farbwerte (RGBA, 0 - 255) aller Pixel der Reihe nach** - je vier Elemente repärsentieren einen Pixel



Und damit lässt sich einiges anstellen …

Alles klar zu Canvas?

4. Webcam-Feeds

User Media API + HTML5-Video

HTML5 Video

```
<video src="videodatei.mp4" controls
  width="640" height="360"></video>
```

User Media API

```
navigator.getUserMedia({
  video: true,
  audio: false
}, function(stream){
  var $video = $('video'), video = $video[0];
  var url = window.URL.createObjectURL(stream);
  $video.attr('src', url);
  video.play();
}, function(err){
  console.error(err);
});
```

Videos sind Canvas-Bildquellen

Sie erinnern sich hieran?

```
// Bild-Objekt erstellen
var img = new Image();
img.src = "test.png";

// Erst zeichnen wenn das Bild geladen ist
img.onload = function(){
  context.drawImage(img, 100, 20);
};
```

Klappt auch mit Video!

Rendering-Loop

```
window.requestAnimationFrame(function loop(){
  // Irgendwas rendern, dann für nächstes Frame:
  if(nichtGestoppt){
    window.requestAnimationFrame(loop);
  }
});
```

requestAnimationFrame() snychronisiert Funktionsaufrufe mit der Browser-Framerate!

… und wie man das ganze dann zusammensetzt, werden Sie dann schon austüfteln.

Alles klar zu Webcam-Capture?

Alles klar?

Noch Fragen zu Drag & Drop, File API, Canvas, HTML5-Video oder User Media API?

Wenn nicht: Pause, dann App-Architektur und JavaScript