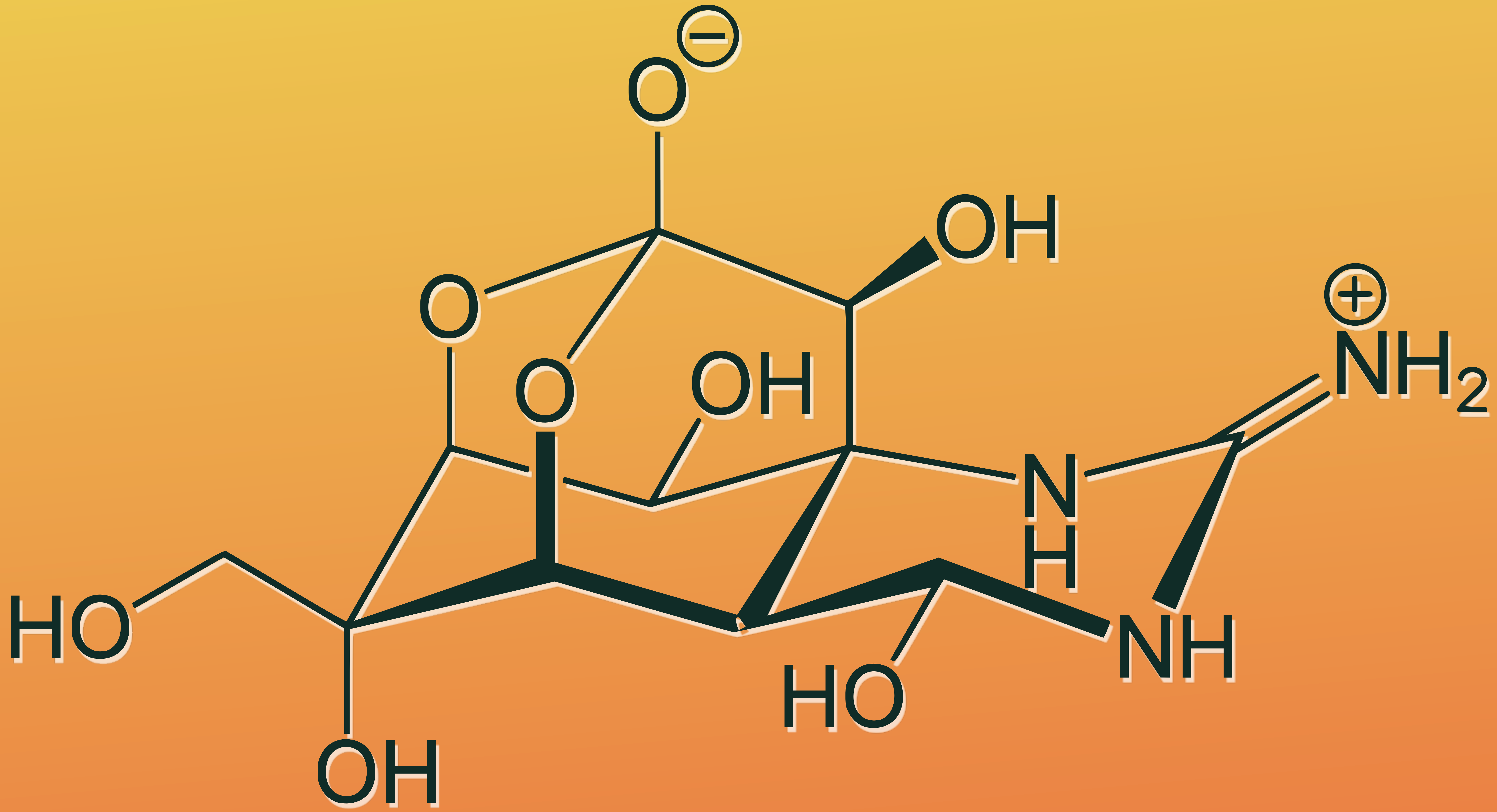
FUGU

POSSIBILITIES WITH WEB CAPABILITIES

DevCon Web & Mobile

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl


O⊖
OH
⊕NH₂
O
O
OH
N H
HO
NH
HO
OH

ROWDY RABOUW

Front-End Focused Senior DevOps Engineer

Google Developer Expert in Web Technologies

Webdeveloper since 1996

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

PROJECT FUGU 🐡

Google Microsoft intel

developer.chrome.com/docs/capabilities

PROJECT FUGU

Accelerometer API	EyeDropper API	Magnetometer API	Web MIDI API
Ambient Light Sensor API	File Handling API	Periodic Background Sync API	Web NFC API
Background Fetch API	File System Access API	Picture-in-Picture API	Web Serial API
Background Sync API	File System Observer API	Presentation API	Web Share API
Badging API	Font Access API	Screen Wake Lock API	Web Speech API
Clipboard API	Gamepad API	Shape Detection API	Web HID API
Compute Pressure API	Geolocation Sensor API	Virtual Keyboard API	Web GPU API
Contact Picker API	Gravity Sensor API	Wake Lock API	Web USB API
Device Memory API	Idle Detection API	Web Audio API	Web XR Device API
Device Posture API	Fullscreen API	Web Bluetooth API	Window Placement API

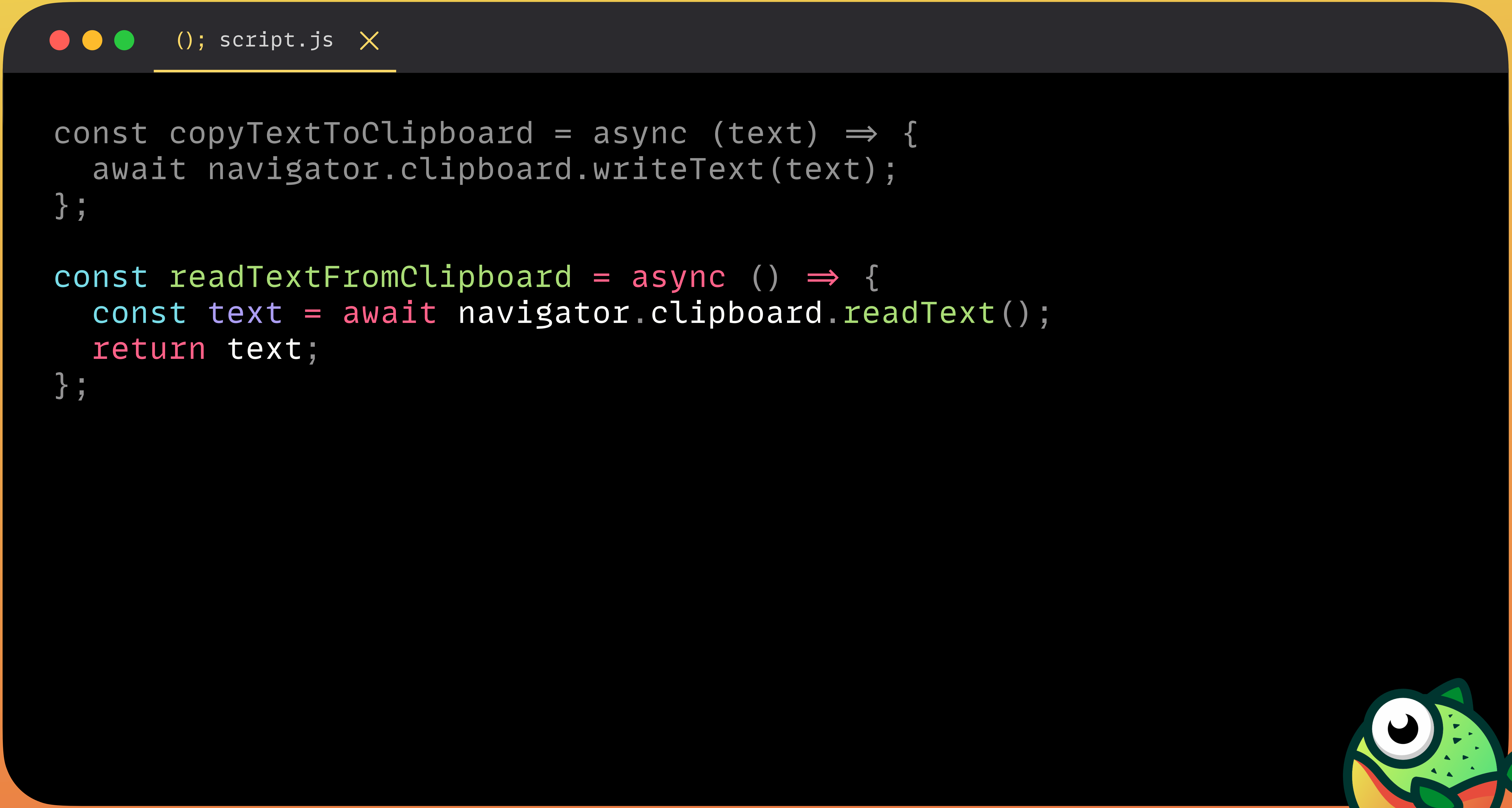
CLIPBOARD API

CLIPBOARD API

- copy text, images, or other data formats to and from the clipboard
- requires a user gesture; for example, a button click
- pasting requires explicit permission from the user

(); script.js

```javascript
const copyTextToClipboard = async (text) => {
  await navigator.clipboard.writeText(text);
};
```



(); script.js

```javascript
const copyTextToClipboard = async (text) => {
  await navigator.clipboard.writeText(text);
};

const readTextFromClipboard = async () => {
  const text = await navigator.clipboard.readText();
  return text;
};
```

(); script.js

```javascript
const copyTextToClipboard = async (text) => {
  await navigator.clipboard.writeText(text);
};

const readTextFromClipboard = async () => {
  const text = await navigator.clipboard.readText();
  return text;
};

const copyPngImageToClipboard = async (imagePath) => {
  const response = await fetch(imagePath);
  const blob = await response.blob();
  const clipboardItem = new ClipboardItem({ "image/png": blob });
  await navigator.clipboard.write([clipboardItem]);
};
```

DEMO

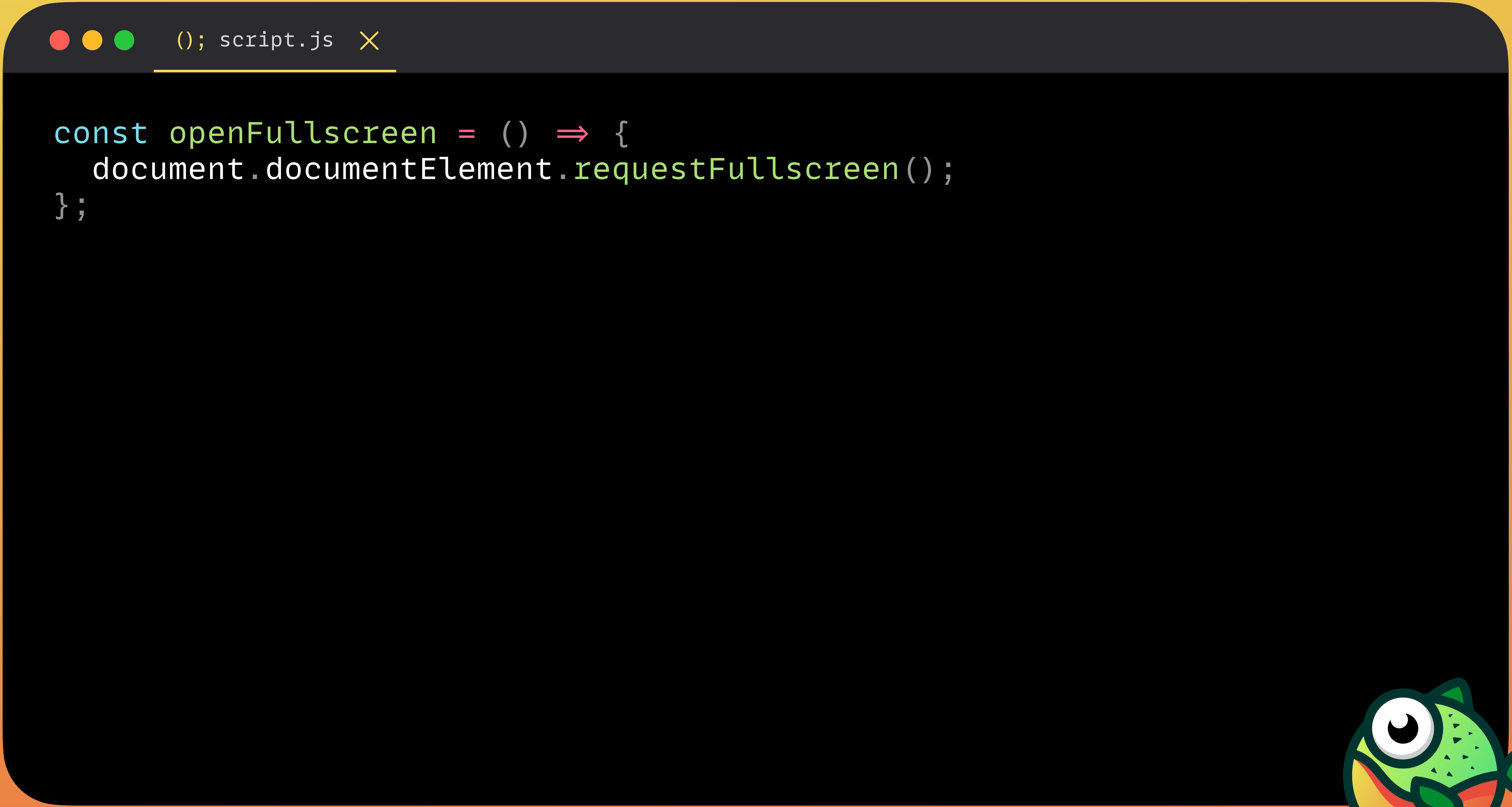
CLIPBOARD API

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

FULLSCREEN API

FULLSCREEN API

- remove distractions like browser UI, so users can focus entirely on the content
- full document or a specific element can be displayed in full-screen mode
- requires a user gesture; for example, a button click



```js
const openFullscreen = () => {
  document.documentElement.requestFullscreen();
};
```

(); script.js

```js
const openFullscreen = () => {
  document.documentElement.requestFullscreen();
};

const exitFullscreen = () => {
  if (document.fullscreenElement) { // prevent error when not fullscreen
    document.exitFullscreen();
  }
};
```

(); script.js

```js
const openFullscreen = () => {
  document.documentElement.requestFullscreen();
};

const exitFullscreen = () => {
  if (document.fullscreenElement) { // prevent error when not fullscreen
    document.exitFullscreen();
  }
};

const video = document.querySelector("#video");

const playVideoFullscreen = (video) => {
  video.requestFullscreen();
  video.play();
};

// user can also exit with Esc key or a button in the video player
```

DEMO

FULLSCREEN API

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

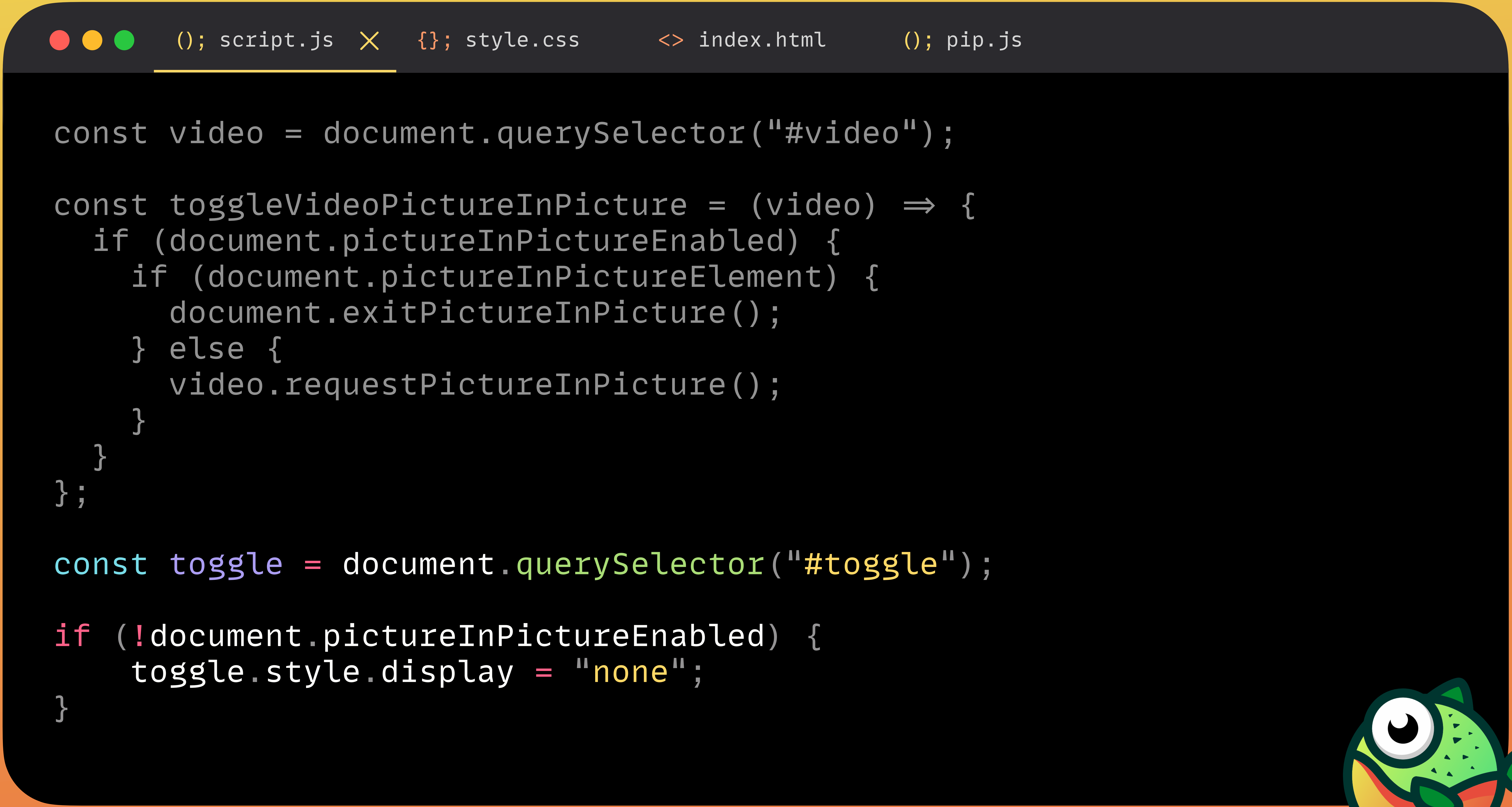
PICTURE-IN-PICTURE API

PICTURE-IN-PICTURE API

- enable a floating, always-on-top frame
- commonly used for video playback, experimental for other content
- requires a user gesture; for example, a button click

(); script.js   {}; style.css   <> index.html   (); pip.js

```js
const video = document.querySelector("#video");

const toggleVideoPictureInPicture = (video) => {
  if (document.pictureInPictureEnabled) {
    if (document.pictureInPictureElement) {
      document.exitPictureInPicture();
    } else {
      video.requestPictureInPicture();
    }
  }
};
```



```js
const video = document.querySelector("#video");

const toggleVideoPictureInPicture = (video) ⇒ {
  if (document.pictureInPictureEnabled) {
    if (document.pictureInPictureElement) {
      document.exitPictureInPicture();
    } else {
      video.requestPictureInPicture();
    }
  }
};

const toggle = document.querySelector("#toggle");

if (!document.pictureInPictureEnabled) {
    toggle.style.display = "none";
}
```

(); script.js   {}; style.css   <> index.html   (); pip.js

```css
:picture-in-picture {
  display: none;
}
```

rowdy.codes   rowdyrabouw   rowdyrabouw   webcapabilities.nl

(); script.js {}; style.css <> index.html (); pip.js

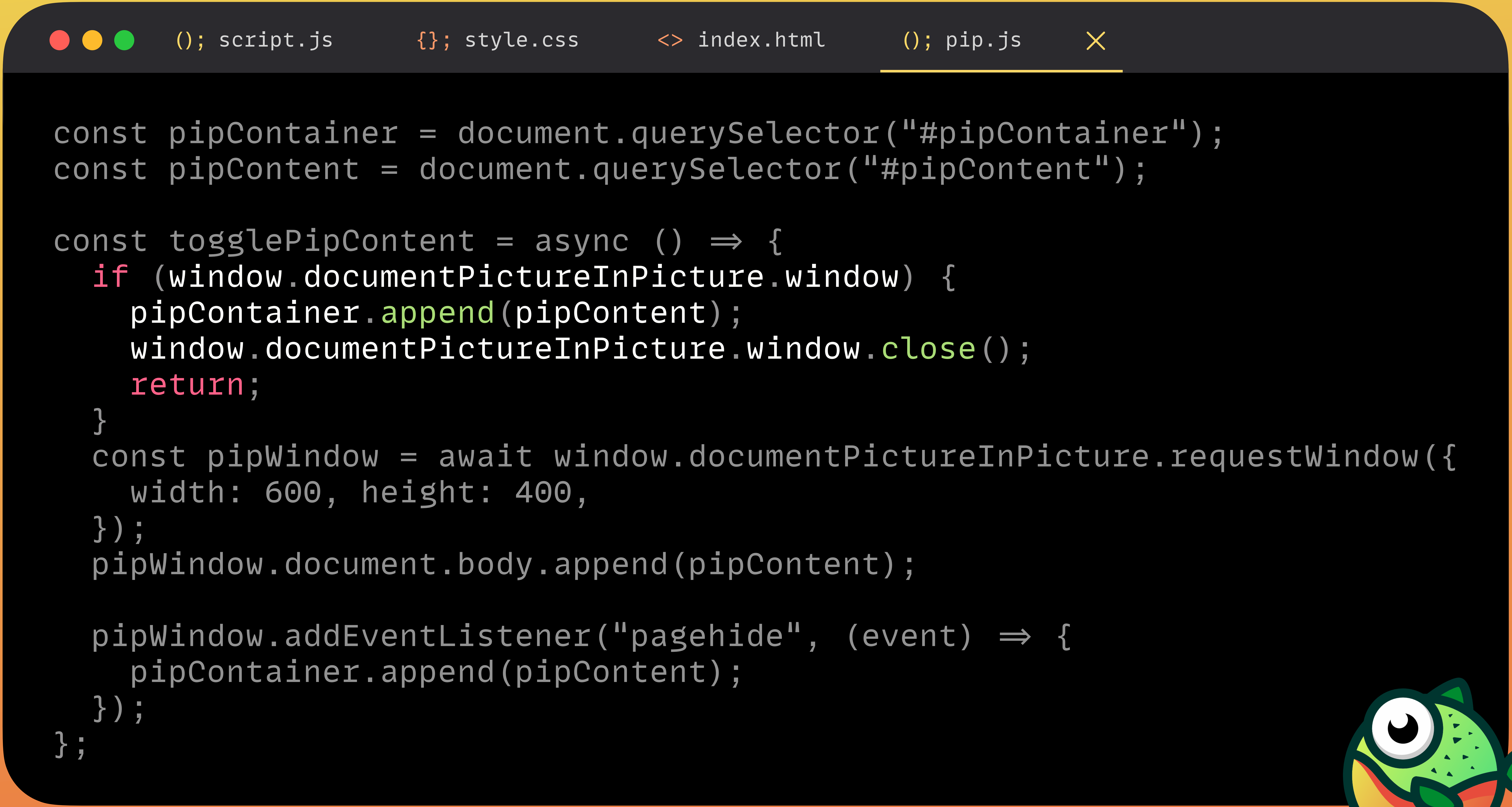
```html
<div id="pipContainer">
  <div id="pipContent">
    <h2>DocumentPictureInPicture</h2>
    <p>This interface allows you to create an always-on-top window.</p>
    <p>This is currently only supported in Chromium based browsers.</p>
  </div>
</div>
```

(); script.js {}; style.css <> index.html (); pip.js

```javascript
const pipContainer = document.querySelector("#pipContainer");
const pipContent = document.querySelector("#pipContent");

const togglePipContent = async () => {





  const pipWindow = await window.documentPictureInPicture.requestWindow({
    width: 600, height: 400,
  });
  pipWindow.document.body.append(pipContent);





};
```

(); script.js   {}; style.css   <> index.html   (); pip.js ✕

```
const pipContainer = document.querySelector("#pipContainer");
const pipContent = document.querySelector("#pipContent");

const togglePipContent = async () ⇒ {





  const pipWindow = await window.documentPictureInPicture.requestWindow({
    width: 600, height: 400,
  });
  pipWindow.document.body.append(pipContent);

  pipWindow.addEventListener("pagehide", (event) ⇒ {
    pipContainer.append(pipContent);
  });
};
```



```js
const pipContainer = document.querySelector("#pipContainer");
const pipContent = document.querySelector("#pipContent");

const togglePipContent = async () => {
  if (window.documentPictureInPicture.window) {
    pipContainer.append(pipContent);
    window.documentPictureInPicture.window.close();
    return;
  }
  const pipWindow = await window.documentPictureInPicture.requestWindow({
    width: 600, height: 400,
  });
  pipWindow.document.body.append(pipContent);

  pipWindow.addEventListener("pagehide", (event) => {
    pipContainer.append(pipContent);
  });
};
```

DEMO

PICTURE-IN-PICTURE API

WEB SHARE API

WEB SHARE API

- share content directly with native sharing capabilities of the user's device
- sharing options depend on the apps installed on the user's device
- limitations on the types of content (audio, video, images, text and pdf)
- requires a user gesture; for example, a button click

(); script.js

```javascript
const shareUrl = async () => {

  if (navigator.canShare) {

    await navigator.share({
      title: "Check out this website!", // may be ignored by the target
      text: "I found this website about project Fugu you should see.",
      url: "https://webcapabilities.nl",
    });

  }

};
```

(); script.js

```javascript
const shareFile = async () ⇒ {

  if (navigator.canShare) {
    const file = new File(["Web Share API", "\n\nThe Web Share API allows
                web applications to share content directly with native sharing
                capabilities of the user's device."]

    );



  }

};
```

(); script.js

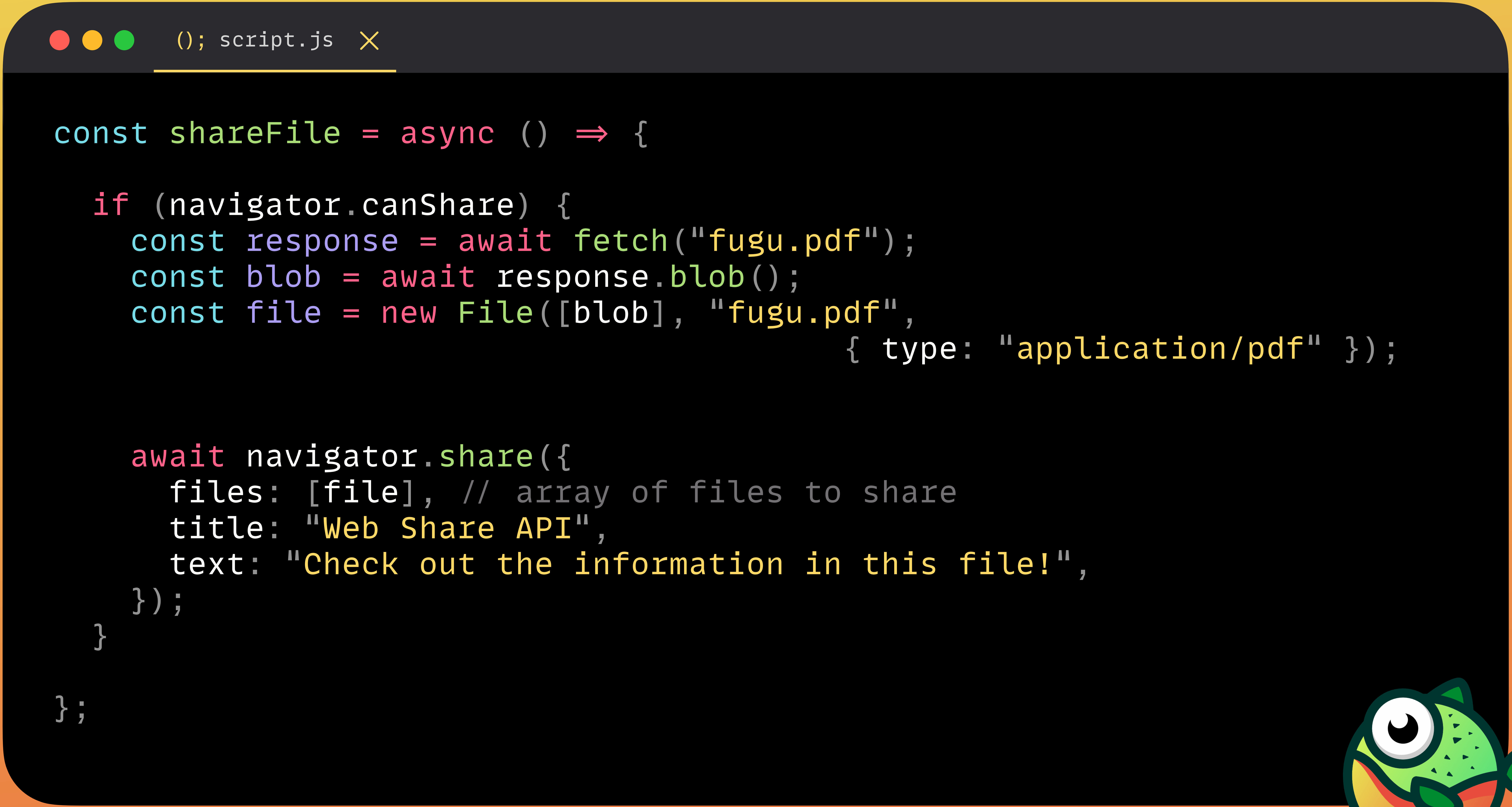
```js
const shareFile = async () => {

  if (navigator.canShare) {
    const file = new File(["Web Share API", "\n\nThe Web Share API allows
               web applications to share content directly with native sharing
               capabilities of the user's device."], "fugu.txt"

    );





  }

};
```

(); script.js

```javascript
const shareFile = async () => {

  if (navigator.canShare) {
    const file = new File(["Web Share API", "\n\nThe Web Share API allows
           web applications to share content directly with native sharing
           capabilities of the user's device."], "fugu.txt",
           { type: "text/plain"}
    );




  }

};
```

(); script.js

```
const shareFile = async () => {

  if (navigator.canShare) {
    const file = new File(["Web Share API", "\n\nThe Web Share API allows
           web applications to share content directly with native sharing
           capabilities of the user's device."], "fugu.txt",
           { type: "text/plain"}
    );

    await navigator.share({
      files: [file], // array of files to share
      title: "Web Share API",
      text: "Check out the information in this file!",
    });
  }

};
```



(); script.js

```javascript
const shareFile = async () => {

  if (navigator.canShare) {
    const response = await fetch("fugu.pdf");
    const blob = await response.blob();
    const file = new File([blob], "fugu.pdf",
                                         { type: "application/pdf" });


    await navigator.share({
      files: [file], // array of files to share
      title: "Web Share API",
      text: "Check out the information in this file!",
    });
  }

};
```

DEMO

WEB SHARE API

WEB USB API

WEB USB API

- communicate directly with USB devices
- connecting requires explicit permission from the user
- requires a user gesture; for example, a button click

(); script.js

```javascript
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {










  }
}
```

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

(); script.js

```javascript
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {
    // select a USB device (hexidecimal values for vendorId and productId)
    const device = await navigator.usb.requestDevice(
        { filters: [{ vendorId: 0x0416, productId: 0x5011 }] });









  }
}
```

(); script.js

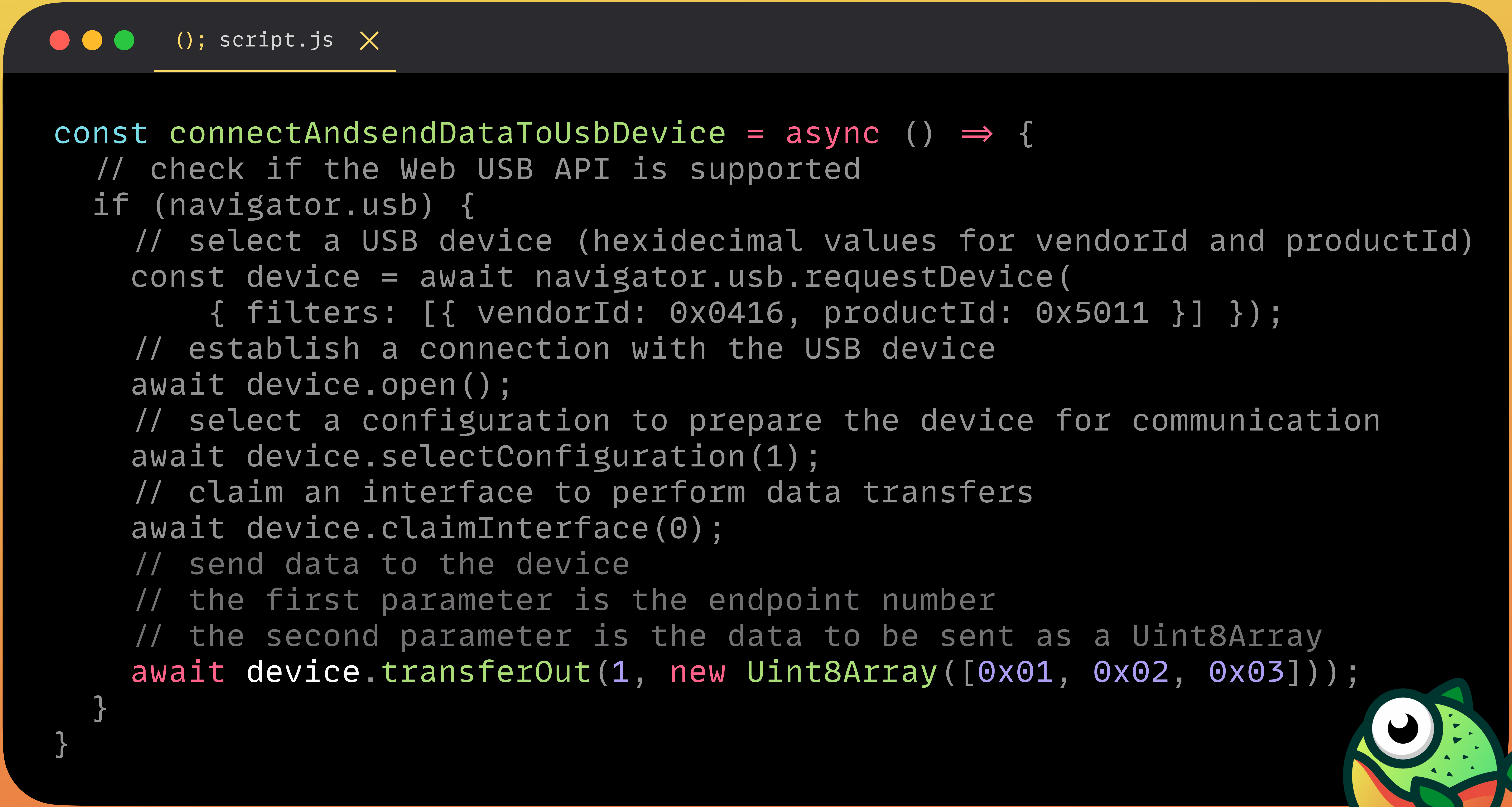
```javascript
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {
    // select a USB device (hexidecimal values for vendorId and productId)
    const device = await navigator.usb.requestDevice(
        { filters: [{ vendorId: 0x0416, productId: 0x5011 }] });
    // establish a connection with the USB device
    await device.open();








  }
}
```

(); script.js

```
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {
    // select a USB device (hexidecimal values for vendorId and productId)
    const device = await navigator.usb.requestDevice(
        { filters: [{ vendorId: 0x0416, productId: 0x5011 }] });
    // establish a connection with the USB device
    await device.open();
    // select a configuration to prepare the device for communication
    await device.selectConfiguration(1);







  }
}
```

(); script.js

```javascript
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {
    // select a USB device (hexidecimal values for vendorId and productId)
    const device = await navigator.usb.requestDevice(
        { filters: [{ vendorId: 0x0416, productId: 0x5011 }] });
    // establish a connection with the USB device
    await device.open();
    // select a configuration to prepare the device for communication
    await device.selectConfiguration(1);
    // claim an interface to perform data transfers
    await device.claimInterface(0);




  }
}
```


(); script.js

```javascript
const connectAndsendDataToUsbDevice = async () => {
  // check if the Web USB API is supported
  if (navigator.usb) {
    // select a USB device (hexidecimal values for vendorId and productId)
    const device = await navigator.usb.requestDevice(
        { filters: [{ vendorId: 0x0416, productId: 0x5011 }] });
    // establish a connection with the USB device
    await device.open();
    // select a configuration to prepare the device for communication
    await device.selectConfiguration(1);
    // claim an interface to perform data transfers
    await device.claimInterface(0);
    // send data to the device
    // the first parameter is the endpoint number
    // the second parameter is the data to be sent as a Uint8Array
    await device.transferOut(1, new Uint8Array([0x01, 0x02, 0x03]));
  }
}
```

DEMO

WEB USB API

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

WIN A FUGU DOG TOY

Send your best tech joke to my printer!

DEMO

WEB USB API

FILE SYSTEM API

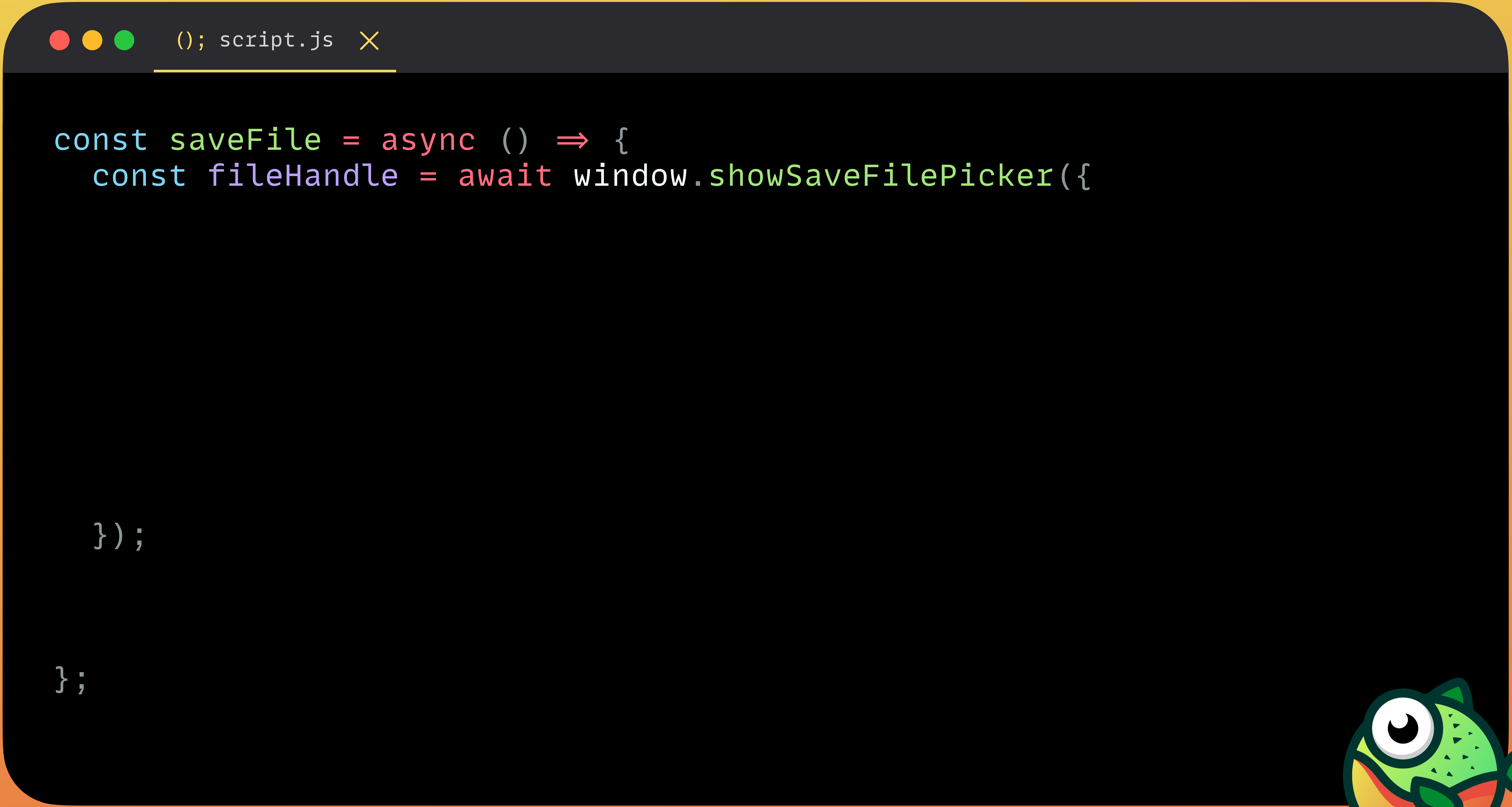
FILE SYSTEM API

- interact with files on the user's local device
- Adobe Photoshop Web
- vscode.dev
- requires a user gesture; for example, a button click

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

(); script.js

```js
const saveFile = async () => {



};
```



(); script.js

```
const saveFile = async () ⇒ {
  const fileHandle = await window.showSaveFilePicker({







  });



};
```

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

(); script.js

```javascript
const saveFile = async () => {
  const fileHandle = await window.showSaveFilePicker({
    types: [
      {
        accept: { 'text/plain': ['.txt'] }
      }
    ]
    startIn: "desktop",
    // FileSystemHandle or known directory ("desktop", "documents",
    // "downloads", "music", "pictures", or "videos")
    suggestedName: "dummy.txt",
  });



};
```

(); script.js

```javascript
const saveFile = async () => {
  const fileHandle = await window.showSaveFilePicker({
    types: [
      {
        accept: { 'text/plain': ['.txt'] }
      }
    ]
    startIn: "desktop",
    // FileSystemHandle or known directory ("desktop", "documents",
    // "downloads", "music", "pictures", or "videos")
    suggestedName: "dummy.txt",
  });
  const writable = await fileHandle.createWritable();


};
```

(); script.js

```javascript
const saveFile = async () => {
  const fileHandle = await window.showSaveFilePicker({
    types: [
      {
        accept: { 'text/plain': ['.txt'] }
      }
    ]
    startIn: "desktop",
    // FileSystemHandle or known directory ("desktop", "documents",
    // "downloads", "music", "pictures", or "videos")
    suggestedName: "dummy.txt",
  });
  const writable = await fileHandle.createWritable();
  await writable.write('Hello World!');
  await writable.close();
};
```

(); script.js

```js
const readFile = async () => {




};
```

(); script.js

```js
const readFile = async () => {
  const fileHandle = await window.showOpenFilePicker(
    {
      types: [
        {
          accept: { 'text/plain': ['.txt'] }
        }
      ]
    });



};
```

(); script.js

```javascript
const readFile = async () => {
  const fileHandle = await window.showOpenFilePicker(
    {
      types: [
        {
          accept: { 'text/plain': ['.txt'] }
        }
      ]
    });
  const file = await fileHandle[0].getFile();


};
```

(); script.js

```javascript
const readFile = async () => {
  const fileHandle = await window.showOpenFilePicker(
    {
      types: [
        {
          accept: { 'text/plain': ['.txt'] }
        }
      ]
    });
  const file = await fileHandle[0].getFile();
  const contents = await file.text();
  console.log(contents);
};
```

DEMO

FILE SYSTEM API

WEB BLUETOOTH API

WEB BLUETOOTH API

- communicate directly with Bluetooth Low Energy (BLE) devices
- devices include fitness trackers, heart rate monitors, and IoT devices
- connecting requires explicit permission from the user
- requires a user gesture; for example, a button click

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl

(); script.js

```javascript
const connectToHeartRateMonitor = async () => {

  // check if the Web Bluetooth API is supported
  if (navigator.bluetooth) {
```

(); script.js

```js
const connectToHeartRateMonitor = async () => {

  // check if the Web Bluetooth API is supported
  if (navigator.bluetooth) {

    // request access to a device
    const device = await navigator.bluetooth.requestDevice({
      filters: [ { namePrefix: "Polar Sense" } ],
      optionalServices: ["heart_rate"],
    });
```

(); script.js

```javascript
const connectToHeartRateMonitor = async () => {

  // check if the Web Bluetooth API is supported
  if (navigator.bluetooth) {

    // request access to a device
    const device = await navigator.bluetooth.requestDevice({
      filters: [ { namePrefix: "Polar Sense" } ],
      optionalServices: ["heart_rate"],
    });

    // connect to the device's Generic Attribute Profile (GATT)
    const server = await device.gatt.connect();
```

(); script.js

```javascript
const connectToHeartRateMonitor = async () => {

  // check if the Web Bluetooth API is supported
  if (navigator.bluetooth) {

    // request access to a device
    const device = await navigator.bluetooth.requestDevice({
      filters: [ { namePrefix: "Polar Sense" } ],
      optionalServices: ["heart_rate"],
    });

    // connect to the device's Generic Attribute Profile (GATT)
    const server = await device.gatt.connect();

    // access the heart rate service
    const service = await server.getPrimaryService("heart_rate");
```

(); script.js

```javascript
    // get the heart rate measurement characteristic
    const characteristic =
               await service.getCharacteristic("heart_rate_measurement");










  }
};
```

(); script.js

```javascript
    // get the heart rate measurement characteristic
    const characteristic =
               await service.getCharacteristic("heart_rate_measurement");

    // enable notifications
    await characteristic.startNotifications();










  }
};
```

(); script.js

```
    // get the heart rate measurement characteristic
    const characteristic =
               await service.getCharacteristic("heart_rate_measurement");

    // enable notifications
    await characteristic.startNotifications();

    // listen for changes of the heart rate
    characteristic.addEventListener("characteristicvaluechanged",
                                                            (event) => {

      const value = event.target.value;
      const heartRate = value.getUint8(1);
      console.log(`Heart Rate: ${heartRate}`);

    });

  }
};
```

Parrot
TAXI

SHARE
OPTIONS

DEMO

WEB BLUETOOTH API
GAMEPAD API

WIN A FUGU DOG TOY

Send your best tech joke to my printer!

MULTUMESC!

ROWDY.CODES/DEVCON

rowdy.codes rowdyrabouw rowdyrabouw webcapabilities.nl