

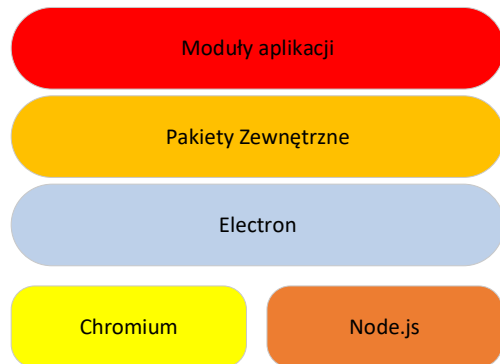
Electron

Procesy electron

Proces główny (Main) - proces uruchamiany w node.js, umożliwia dostęp do zasobów systemowych (np. system plików)

Proces okna (Renderer) - uruchamiany w chrome (możliwości podobne do tego jakie daje przeglądarka)

Budowa aplikacji Electron









Tworzenie projektu

```
npm init -y  
npm i -D electron@latest
```

Tworzenie projektu

Struktura plików

	Nazwa elementu	Opis
 node_modules	node_modules	Folder przechowujący zewnętrzne biblioteki
 index.html	index.html	Plik zawierający wygląd graficzny aplikacji
 main.js	main.js	skrypt stanowi punkt wejściowy do stworzonej aplikacji - zawiera kod głównego procesu. Zawarte w nim operacje mają na celu zarządzanie cyklem życia aplikacji, wykonywanie natywnych operacji w systemie (aplikacja może posiadać wyłącznie jeden główny proces)
 package.json	package.json	plik konfiguracyjny aplikacji (informacje na temat projektu, komendy uruchamiające aplikację)
 package-lock.json	package-lock.json	
 preload.js	preload.js	pomost umożliwiający wykonywanie funkcjonalności Node.js z poziomu kodu zawartego w interfejsie graficznym

Tworzenie projektu

preload.js:

```
const { contextBridge } = require('electron');
const fs = require('fs');

contextBridge.exposeInMainWorld('api', {

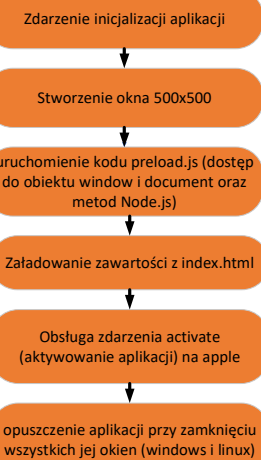
  readFile: function (filePath) {
    return fs.readFileSync(filePath, 'utf8');
  }

});
```

W pliku powinna się znaleźć funkcjonalność Node.js udostępniona do wykorzystania w skryptach przeglądarki internetowej.

(elementy będą widoczne w obiekcie **api** obiektu **window**).

```
const { app, BrowserWindow } = require('electron')
const path = require('path')
function createWindow() {
  const window = new BrowserWindow({
    width: 500, height: 500,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js')
    }
  })
  window.loadFile('index.html')
}
app.whenReady().then(() => {
  createWindow()
  app.on('activate', () => {
    if (BrowserWindow.getAllWindows().length === 0) {
      createWindow()
    }
  })
})
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit()
  }
})
```

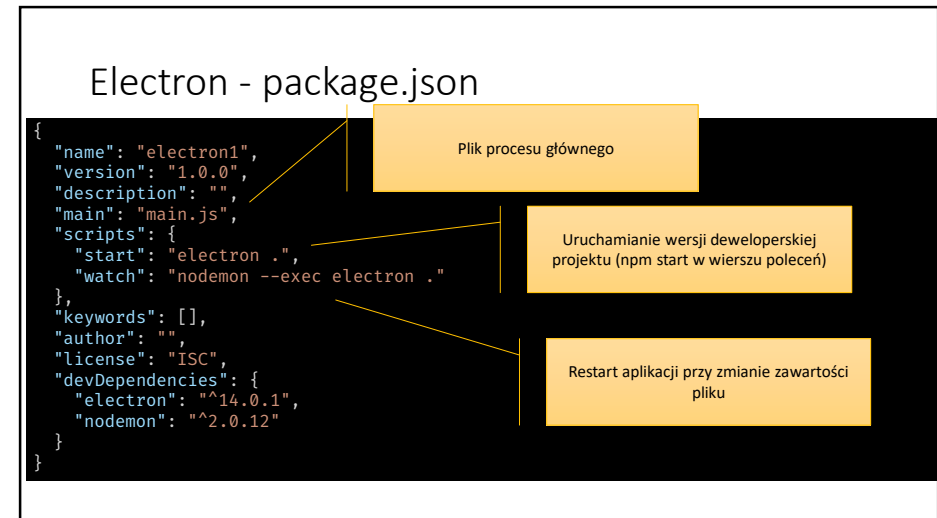
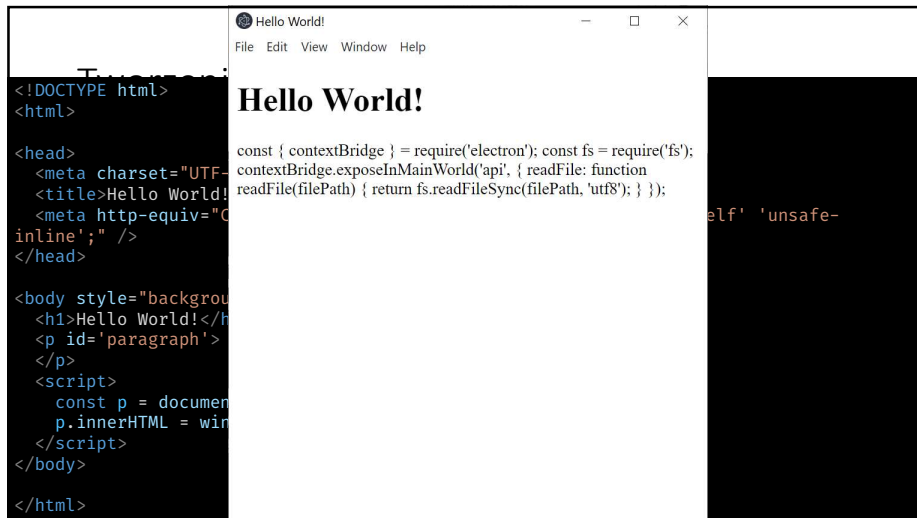


```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Hello World!</title>
  <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline';" />
</head>

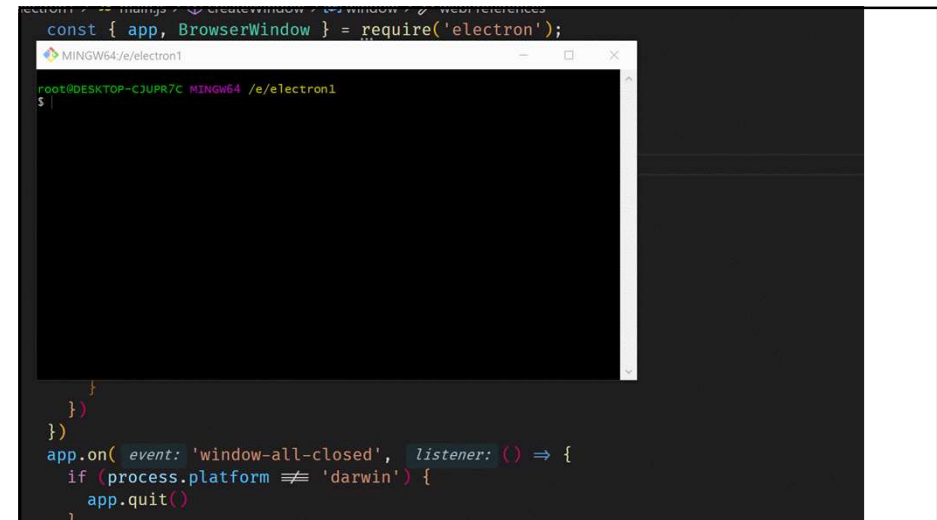
<body style="background: white;">
  <h1>Hello World!</h1>
  <p id='paragraph'>
  </p>
  <script>
    const p = document.getElementById('paragraph');
    p.innerHTML = window.api.readFile('./preload.js');
  </script>
</body>

</html>
```



Electron - package.json

```
npm init -y
npm i -D electron@latest
npm i -D nodemon
```



```
ron1 > JS main.js > createWindow > window > width
const { app, BrowserWindow } = require('electron');

MINGW64/e/electron1

root@DESKTOP-CJUPR7C MINGW64 /e/electron1
: |

}
})
app.on( event: 'window-all-closed', listener: () => {
  if (process.platform !== 'darwin') {
    app.quit()
  }
})
```

Tworzenie projektu

app - wybrane metody:

Metoda	Opis
quit	Próbuje zamknąć wszystkie okna. Akcja może się nie powieść gdy w którejkolwiek metodzie beforeunload zostanie zwrócona wartość false.
exit	Natychmiast zamyka aplikację
isReady	true gdy inicjalizacja została zakończona
whenReady	Zwraca Promise, która przyjmuje stan fulfilled gdy aplikacja zostanie zainicjalizowana
focus	aktywuje okno przeglądarki
getAppPath	zwraca ścieżkę folderu aplikacji
getPath(string)	zwraca ścieżkę do danego elementu w systemie (np. do folderu dokumentów)

Tworzenie projektu

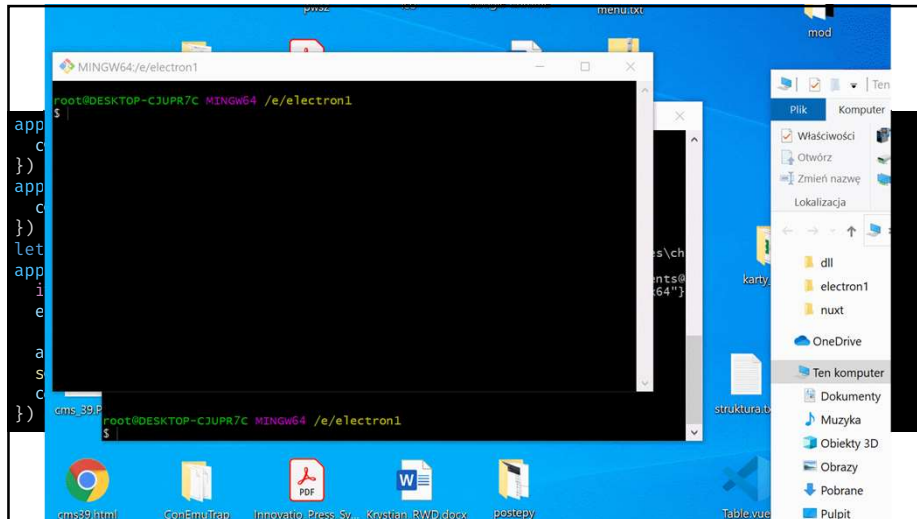
app - element zarządza cyklem życia aplikacji.

Przykładowe:

Zdarzenie	Opis
ready	Emitowane po zakończeniu inicjalizacji aplikacji.
before-quit	Emitowane przed zamknięciem okna aplikacji
window-all-closed	Emitowane w przypadku zamknięcia wszystkich okien aplikacji
browser-window-blur	Emitowane w przypadku, gdy aplikacja przestaje być aktywna
browser-window-focus	Emitowane przy aktywacji okna przeglądarki

```
app.on('browser-window-blur', e=>{
  console.log('Okno nieaktywne');
})
app.on('browser-window-focus', e=>{
  console.log('Okno aktywne');
})
let appDataSaved = false;
app.on('before-quit', e=>{
  if(!appDataSaved)
    e.preventDefault();

  appDataSaved = true;
  setTimeout(()=>app.quit(), 3000);
  console.log('Za 3 s aplikacja zostanie zamknięta');
})
```



Tworzenie projektu

BrowserWindow - wybrane zdarzenia:

Zdarzenie	Opis
close	przy zamykaniu okna (przed beforeunload i unload) - wywołanie preventDefault() anuluje operację
closed	po zamknięciu okna
blur, focus	zdarzenia przy aktywacji / dezaktywacji okna
show, hide	przy pokazywaniu / chowaniu okna
ready-to-show	wywoływane w chwili, gdy okno jest gotowe do wyświetlenia (bez migania)
maximize, unmaximize, minimize, restore	przy podstawowych operacjach zmieniających rozmiar
resize	po zmianie rozmiaru
move	przy przeciąganiu

Tworzenie projektu

BrowserWindow

Element odpowiedzialny za tworzenie i zarządzaniem oknem przeglądarki. Wybrane parametry przekazywane w obiekcie do konstruktora ({ ... }):

Parametr	Opis
width, height, x, y	położenie i rozmiar okna
center	środkuje okno (dla true)
minWidth, minHeight, maxWidth, maxHeight	ograniczenia rozmiaru okna
resizable, minimizable, maximizable, closable, focusable	Czy okno można zminimalizować, zmaksymalizować, zamknąć, aktywować, zmienić rozmiar
title, icon	tytuł, ikonka okna
show	czy okno ma zostać pokazane po utworzeniu
frame	czy pokazywać ramkę okna
parent	okno nadrzędne
modal	czy okno modalne
webPreferences	dodatkowy obiekt ustawień

Tworzenie projektu

BrowserWindow - wybrane metody statyczne:

Zdarzenie	Opis
getAllWindows	zwraca tablicę otwartych okien
getFocusedWindow	aktywne okno
fromId	zwraca okno o przekazanym identyfikatorze

Tworzenie projektu

BrowserWindow - właściwości:

Zdarzenie	Opis
webContents (tylko do odczytu)	zwraca obiekt WebContents powiązany z oknem
id (tylko do odczytu)	identyfikator
fullScreen	czy okno jest powiększone na cały ekran
title	tytuł
maximizable, fullScreenable, resizable, closable, movable	właściwości logiczne określające możliwości operowania na oknie

Tworzenie projektu

BrowserWindow - wybrane metody obiektu:

Zdarzenie	Opis
setMinimumSize / getMinimumSize	ustawia / pobiera minimalny rozmiar okna
setMaximumSize / getMaximumSize	ustawia / pobiera maksymalny rozmiar okna
setResizable / isResizable, setFullScreenable / isFullScreenable	ustawia / pobiera wartość logiczną czy okno jest możliwe do zmiany rozmiaru / wyświetlenia na całym ekranie
moveTop	ustawia okno na pierwszym planie
center	ustawia na środku ekranu
setPosition / getPosition	ustawia / pobiera pozycję okna
setTitle / getTitle	ustawia / pobiera tytuł
capturePage([rect])	tworzy zrzut ekranu z okna (parametr opcjonalny pozwala określić lokalizację zrzutu)
loadURL	ładuje zawartość okna z adresu URL. (powiązane zdarzenia did-finish-load i did-fail-load)

Tworzenie projektu

Zdarzenie	Opis
destroy	wymusza zamknięcie okna. Zdarzenia unload, beforeunload, close nie zostaną wywołane. Zdarzenie closed BĘDZIE
close	próbuję zamknąć okno.
focus, blur	aktywuje / dezaktywuje okno
isFocused / isDestroyed	czy okno jest aktywne / zniszczone
show	wyświetla i aktywuje okno
showInactive	wyświetla i dezaktywuje okno
hide	chowa okno
isVisible, isModal	czy okno jest widoczne / czy to okno modalne
maximize, unmaximize, isMaximized, minimize, restore, isMinimized, isFullScreen, isFullScreenable	zbiór metod zarządzających rozmiarem i widocznością okna
setBounds({x:, y:, width:, height:})	ustala lokalizację okna
getBounds	zwraca lokalizację
setSize / getContentSize	ustawia / zwraca rozmiar okna

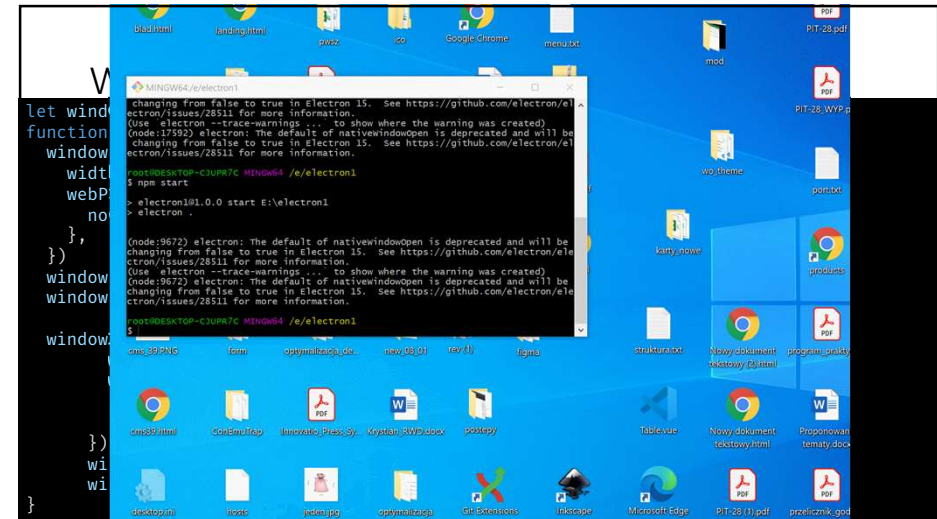
Tworzenie projektu

BrowserWindow - wybrane metody obiektu:

Zdarzenie	Opis
loadFile	ładuje zawartość okna z pliku (did-finish-load, did-fail-load)
setMenu (linux, Windows)	ustawia Menu główne
removeMenu (linux, Windows)	usuwa Menu główne
setIcon	ustawia ikonę okna
setParentWindow	ustawia okno rodzicielskie
getChildWindows	pobiera okna potomne

Wyświetlanie okna po załadowaniu zawartości

```
function createWindow() {
  const window = new BrowserWindow({
    width: 1100,      height: 500,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    },
    show: false,
  })
  window.loadFile('index.html');
  window.once('ready-to-show', window.show);
}
```



Wiele NIEZALEŻNYCH okien

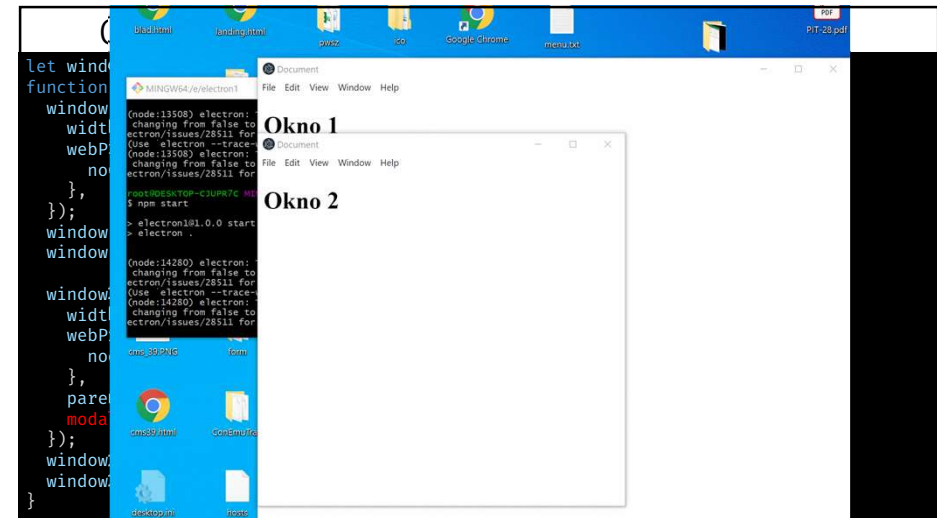
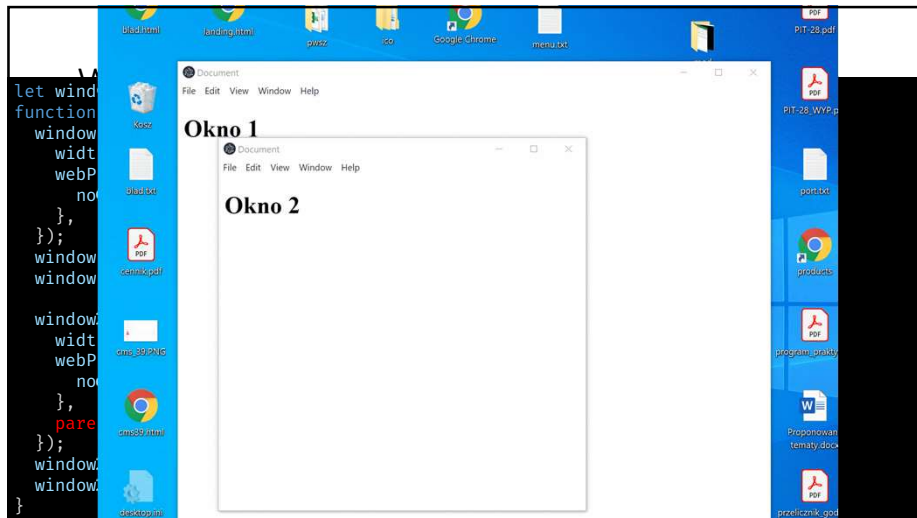
```
let window, window2;
function createWindow() {
  window = new BrowserWindow({
    width: 800,      height: 800, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window2 = new BrowserWindow({
    width: 500,      height: 500, y: 200, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window2.loadFile('index2.html');
  window2.on('closed', ()=>{ window = null;});
}
```

Wiele ZALEŻNYCH okien

```
let window, window2;
function createWindow() {
  window = new BrowserWindow({
    width: 800,      height: 800, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window2 = new BrowserWindow({
    width: 500,      height: 500, y: 200, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
    parent: window,
  });
  window2.loadFile('index2.html');
  window2.on('closed', ()=>{ window = null;});
}
```



Okno modalne

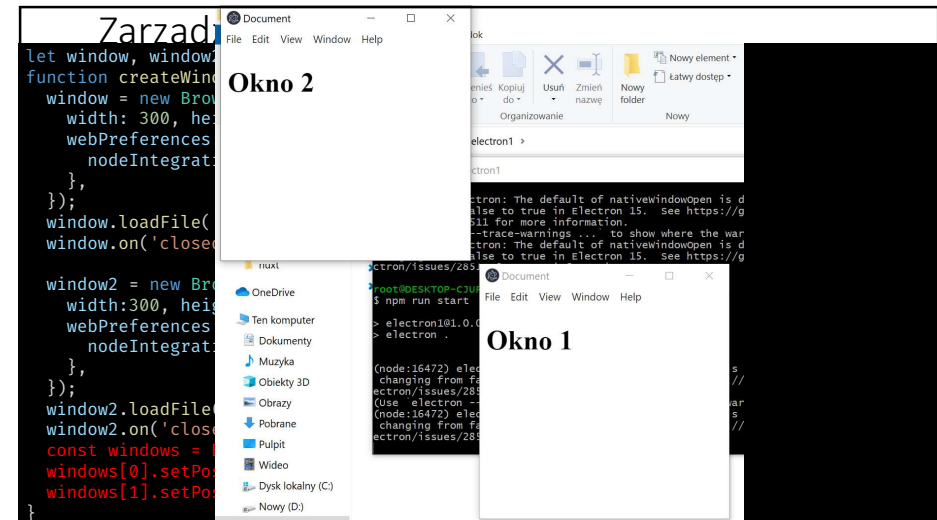
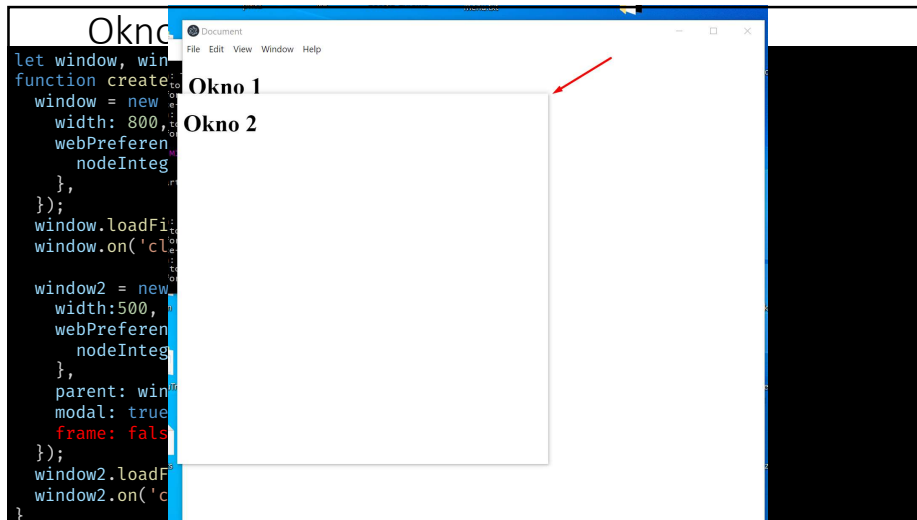
```
let window, window2;
function createWindow() {
  window = new BrowserWindow({
    width: 800,      height: 800, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window2 = new BrowserWindow({
    width:500,      height: 500, y: 200, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
    parent: window,
    modal: true,
  });
  window2.loadFile('index2.html');
  window2.on('closed', ()=>{ window = null;});
}
```

Okno bez obramowania

```
let window, window2;
function createWindow() {
  window = new BrowserWindow({
    width: 800,      height: 800, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window2 = new BrowserWindow({
    width:500,      height: 500, y: 200, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
    parent: window,
    modal: true,
    frame: false,
  });
  window2.loadFile('index2.html');
  window2.on('closed', ()=>{ window = null;});
}
```

Zarządzanie oknami

webContents

webContents
element związany z zawartością okna. Wybrane metody:

Metoda	Opis
<code>getAllWebContents</code>	pobiera tablicę obiektów WebContents dla wszystkich okien i innych obiektów związanych z procesem głównym
<code>getFocusedWebContents</code>	pobiera obiekt związany z aktywnym elementem
<code>fromId</code>	pobiera obiekt związany z określonym id

webContents

webContents

wybrane zdarzenia:

Zdarzenie	Opis
did-finish-load	emitowane, gdy nawigacja została zakończona.
did-fail-load	gdy ładowanie zawartości nie powiodło się
did-start-loading, did-stop-loading	początek ładowania / koniec ładowania zawartości
dom-ready	gdy dokument jest załadowany
will-navigate	gdy użytkownik chce zacząć nawigować do innego źródła (przy zmianie obiektu window.location)
unresponsive, responsive	gdy strona nie jest / jest możliwa do interakcji z użytkownikiem
destroyed	gdy webContents został zniszczony
before-input-event	przed zdarzeniami klawiatury
ipc-message	gdy renderer wysła wiadomość przez ipcRenderer.send

webContents

webContents

wybrane metody:

Zdarzenie	Opis
sendInputEvent	wysła zdarzenie myszy, klawiatury do strony
savePage	zapisuje stronę
invalidate	pełne odrysowanie strony

webContents

webContents

wybrane metody:

Zdarzenie	Opis
loadURL	ładuje zawartość z adresu URL
loadFile	ładuje zawartość z pliku
downloadURL	Inicjalizuje pobieranie elementu bez nawigacji do niego
getURL	zwraca URL bieżącej strony
getTitle	pobiera tytuł strony
isDestroyed, isFocused, isLoading	metody informacyjne o stanie strony
focus, stop, reload	metody wymuszające aktywność, zatrzymanie, przeładowanie
executeJavaScript	wywołuje kod JS i zwraca rezultat wynikający z kodu
undo, redo, cut, copy, copyImageAt(x,y), paste, delete, selectAll, unselect	podstawowe metody edycji
insertText	umieszcza tekst w aktywnym elemencie
capturePage	tworzy zrzut ekranu ze strony

webContents

```
const { app, BrowserWindow, webContents } = require('electron');
const path = require('path');

let window;
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
};
window.loadFile('index.html');
window.on('closed', ()=>{ window = null;});

let windowWebContents = window.webContents;
windowWebContents.on('before-input-event', (e, data)=>{
  console.log(`${data.key} - ${data.type}`);
});
}
```

webContents

The screenshot shows a window titled "Okno 1" with a dark background. Below the window, a terminal window is open, showing the command prompt for an Electron application. The terminal output includes several warnings about the deprecated `nativeWindowOpen` property and the command `npm run start` being executed. The application is running in a MINGW64 environment.

```
let window;
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  let windowWebContents = window.webContents;
  window.on('blur', (e)=>{
    windowWebContents.executeJavaScript(
      'document.querySelector("#header").innerHTML="okno nieaktywne"'
    );
  });

  window.on('focus', (e)=>{
    windowWebContents.executeJavaScript(
      'document.querySelector("#header").innerHTML="okno aktywne"'
    );
  });
}
```

webContents

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1 id="header"> Okno 1</h1>
  <script>

  </script>
</body>
</html>
```

The screenshot shows a window titled "okno nieaktywne" with a dark background. Below the window, a terminal window is open, showing the command prompt for an Electron application. The terminal output includes several warnings about the deprecated `nativeWindowOpen` property and the command `npm run start` being executed. The application is running in a MINGW64 environment. The window content shows a list of keyboard events (keyUp, keyDown) and a message about a network service crash.

session

session

element związane ze stanem webContents okna (cache, ciasteczka, localStorage, IndexedDB).

Sesja domyślna jest dzielona pomiędzy wszystkimi oknami aplikacji.

Element session obiektu webContents zwraca obiekt związany z sesją.

Do okna można przypisać oddzielną sesję podając jej nazwę w atrybucie partition w webPreferences konstruktora BrowserWindow.

Rozpoczynając nazwę od wyrażenia "persist:" sesja będzie utrwalana (wartości będą pamiętane pomiędzy restartami aplikacji).

Domyślna sesja jest automatycznie utrwalana.

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
</head>
<body>
  <script>
    let window;
    function createWindow() {
      window = new BrowserWindow({
        width: 300, height: 300,
        webPreferences: {
          nodeIntegration: true,
        },
      });
      window.loadFile('index.html');
      window.on('closed', () => {
        // Window closed.
      });
    }
    let windowWebContents;
    windowWebContents = window.webContents;
    windowWebContents.executeJavaScript(
    </script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1 id="header"> Okno 1</h1>
  <button id="btn">Odczytaj wartość</button>
  <button id="btn2">Ustaw wartość</button>
  <script>
    let button = document.querySelector("#btn");
    button.addEventListener("click", (e) => {
      let header = document.querySelector("#header");
      header.innerHTML = localStorage.getItem("data1");
    });
    let button2 = document.querySelector("#btn2");
    button2.addEventListener("click", (e) => {
      localStorage.setItem("data1", "Dane związane z kluczem data1 " + Math.random());
    });
  </script>
</body>
</html>
```

session

```
window = new BrowserWindow({
  width: 300, height: 300, y: 100, x: 300,
  webPreferences: {
    nodeIntegration: true, contextIsolation: false,
    partition: "partition1"
  },
});
```

session

session

```

window = new BrowserWindow({
  width: 300, height: 300,
  webPreferences: {
    nodeIntegration: true,
    partition: "persist:partition1"
  },
});

```

session

session

```

window = new BrowserWindow({
  width: 300, height: 300,
  webPreferences: {
    nodeIntegration: true,
    partition: "persist:partition1"
  },
});

```

session

```

window = new BrowserWindow({
  width: 300, height: 300, y: 100, x: 300,
  webPreferences: {
    nodeIntegration: true, contextIsolation: false,
    partition: "persist:partition1"
  },
});

```

dialog

dialog

zarządza natywnymi oknami dialogowymi systemu operacyjnego

Metoda	Opis
<code>showOpenDialogSync, showOpenDialog</code>	Wyświetla okno modalne wyboru pliku/katalogu
<code>showSaveDialogSync, showSaveDialog</code>	okno zapisu pliku
<code>showMessageBoxSync, showMessageBox</code>	informacyjne

1. `browserWindow` - obiekt, dla którego ma być wyświetlone okno modalne (okno rodzicielskie)
2. `options`: `title` (tytuł), `defaultPath` (domyślna ścieżka), `buttonLabel` (etykieta przycisku potwierdzenia) ...

dialog

```
const { app, BrowserWindow, dialog } = require('electron');
const path = require('path');

let window;

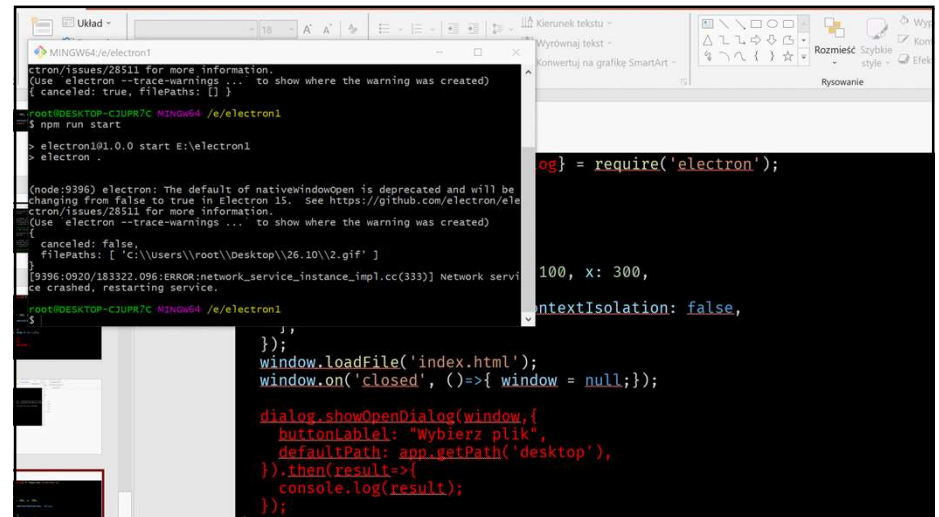
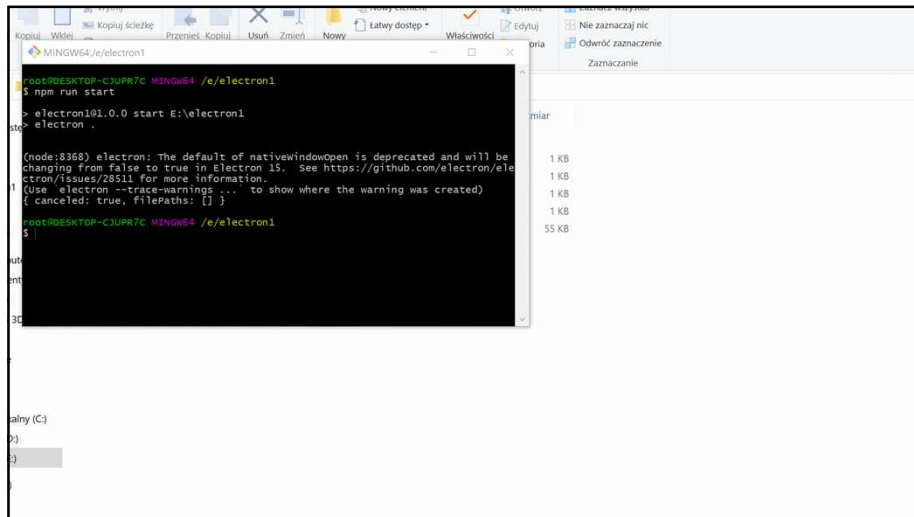
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  dialog.showOpenDialog(window,{
    buttonLabel: "Wybierz plik",
    defaultPath: app.getPath('desktop'),
  }).then(result=>{
    console.log(result);
  });
}
```

dialog

```
let window;
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  const labels = ["C++", "JavaScript", "C#", "Java", "Assembler"];
  dialog.showMessageBox({
    title: "Wybór języku programowania",
    message: "W którym języku programowania programujesz najlepiej:",
    buttons: labels,
  }).then(result=>{
    console.log(result);
  })
}
```



Menu, MenuItem

Menu

główne menu aplikacji

Metoda Statyczna	Opis
<code>Menu.setApplicationMenu(menu)</code>	przypisuje menu do aplikacji
<code>Menu.getApplicationMenu()</code>	pobiera menu aplikacji (obiekt)
<code>Menu.buildFromTemplate(template)</code>	tworzy menu na podstawie szablonu

Menu, MenuItem

Zdarzenia Menu

Zdarzenie	Opis
<code>menu-will-show</code>	emitowane przy wywołaniu metody <code>popup()</code>
<code>menu-will-close</code>	emitowane przy zamykaniu menu manualnie lub przy wywołaniu metody <code>closePopup()</code>
<code>append(menuItem)</code>	dodaje pozycję menu
<code>getMenuItemById(id)</code>	zwraca MenuItem z określonym id
<code>insert(pozycja, menuItem)</code>	umieszcza pozycję w określonym miejscu menu

Menu, MenuItem

Menu

Metoda	Opis
<code>popup</code>	wyświetla menu jako kontekstowe
<code>closePopup</code>	zamyka menu kontekstowe
<code>append(menuItem)</code>	dodaje pozycję menu
<code>getMenuItemById(id)</code>	zwraca MenuItem z określonym id
<code>insert(pozycja, menuItem)</code>	umieszcza pozycję w określonym miejscu menu

Menu, MenuItem

MenuItem

wybrane elementy:

Wybrane elementy MenuItem	Opis
<code>id</code>	String określający identyfikator
<code>label</code>	etykieta
<code>click</code>	funkcja wywoływana przy kliknięciu elementu menu
<code>submenu</code>	obiekt Menu, gdy element zawiera menu zagnieżdżone
<code>type</code>	normal, separator, submenu, checkbox, radio

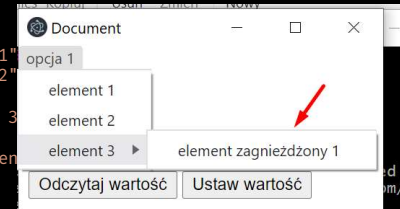
```
const { app, BrowserWindow, Menu, MenuItem } = require('electron');
const path = require('path');
let window;
let menu = new Menu();
let menuItem1 = new MenuItem(
  {
    label: 'opcja 1',
    submenu: [
      {label: "element 1"},
      {label: "element 2"},
      {label: "element 3"},
    ]
  }
);
menu.append(menuItem1);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {nodeIntegration: true, contextIsolation: false},
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});
}
Menu.setApplicationMenu(menu);
}
```

```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: 'opcja 1',
    submenu: [
      {label: "element 1"},
      {label: "element 2"},
      {
        label: "element 3",
        submenu: [
          {label: "element zagnieżdżony 1"}
        ]
      }
    ]
  }
]);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});
}
```

```
const { app, BrowserWindow, Menu, MenuItem } = require('electron');
const path = require('path');
let window;
let menu = new Menu();
let menuItem1 = new MenuItem(
  {
    label: 'opcja 1',
    submenu: [
      {label: "element 1"},
      {label: "element 2"},
      {label: "element 3"},
    ]
  }
);
menu.append(menuItem1);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {nodeIntegration: true, contextIsolation: false},
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});
}
Menu.setApplicationMenu(menu);
}
```



```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: 'opcja 1',
    submenu: [
      {label: "element 1"},
      {label: "element 2"},
      {
        label: "element 3",
        submenu: [
          {label: "element zagnieżdżony 1"}
        ]
      }
    ]
  }
]);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});
}
```



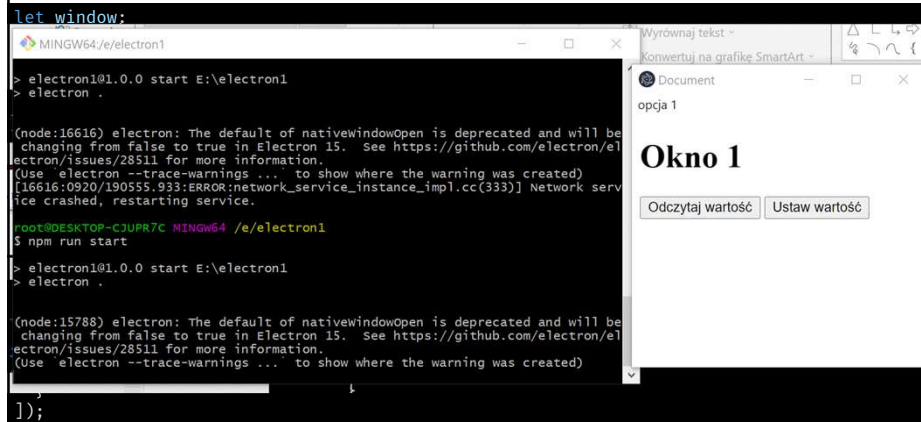
Menu, MenuItem

```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: 'opcja 1',
    submenu: [
      {
        label: "element 1",
        click: ()=>{ console.log('Element 1 kliknięty'); }
      },
      {
        label: "element 2",
        click: ()=>{ console.log('Element 2 kliknięty'); }
      }
    ]
  }
]);
```

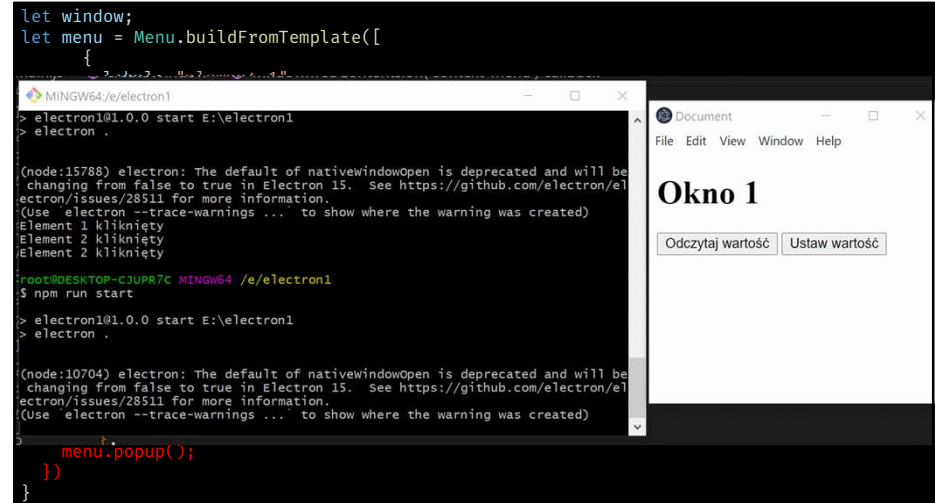
```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: "element 1",
    click: ()=>{ console.log('Element 1 kliknięty'); }
  },
  {
    label: "element 2",
    click: ()=>{ console.log('Element 2 kliknięty'); }
  }
]);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: { nodeIntegration: true, contextIsolation: false, },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window.webContents.on('context-menu', e=>{
    menu.popup();
  })
}
```

Menu, MenuItem



```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: 'opcja 1',
    submenu: [
      {
        label: "element 1",
        click: ()=>{ console.log('Element 1 kliknięty'); }
      },
      {
        label: "element 2",
        click: ()=>{ console.log('Element 2 kliknięty'); }
      }
    ]
  }
]);
```



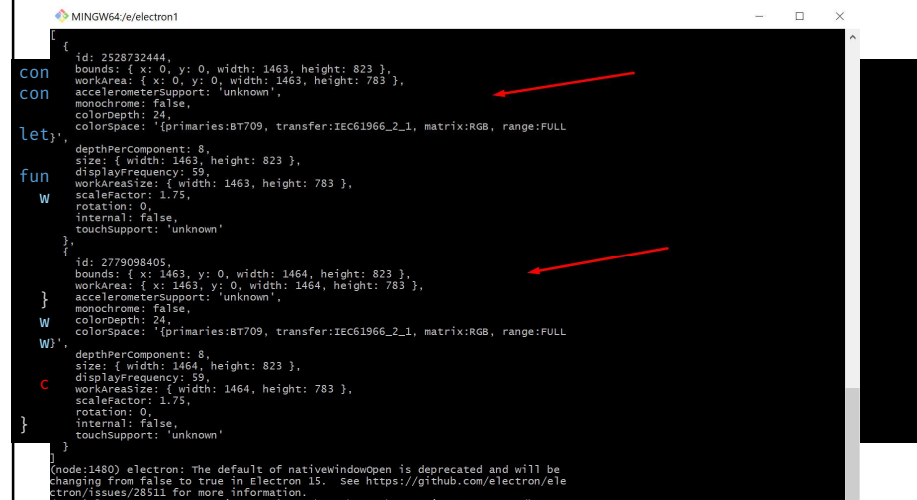
```
let window;
let menu = Menu.buildFromTemplate([
  {
    label: "element 1",
    click: ()=>{ console.log('Element 1 kliknięty'); }
  },
  {
    label: "element 2",
    click: ()=>{ console.log('Element 2 kliknięty'); }
  }
]);
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: { nodeIntegration: true, contextIsolation: false, },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  window.webContents.on('context-menu', e=>{
    menu.popup();
  })
}
```

screen

Wybrane elementy screen

<code>getCursorScreenPoint()</code>	absolutna pozycja kursora (punkt)
<code>getPrimaryDisplay()</code>	obiekt zawierający informacje o głównym ekranie
<code>primaryDisplay.workAreaSize</code>	informacje na temat szerokości i wysokości ekranu



```
const { app, BrowserWindow, screen } = require('electron');
const path = require('path');

let window;

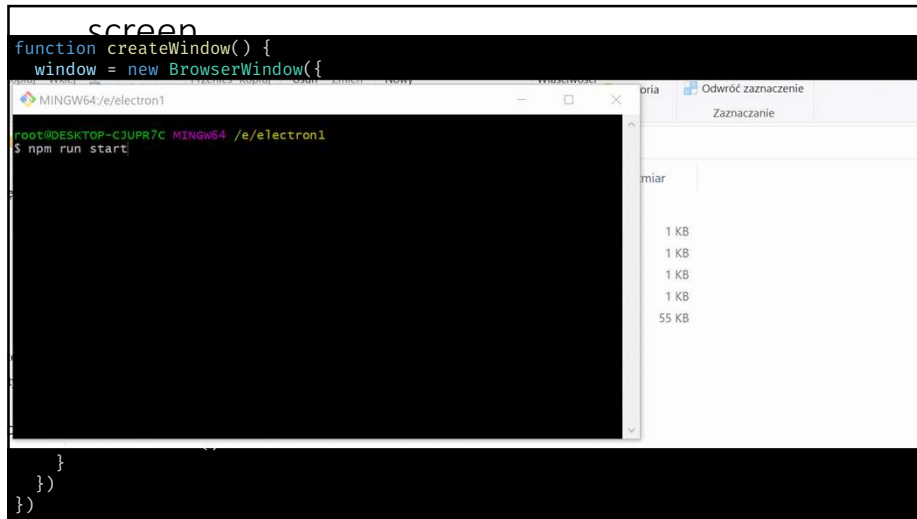
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  console.log(screen.getAllDisplays());
}
```

```
function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', ()=>{ window = null;});

  setInterval(()=>{
    console.log(screen.getCursorScreenPoint());
  }, 500);
}

app.whenReady().then(() => {
  createWindow();
  app.on('activate', () => {
    if (BrowserWindow.getAllWindows().length === 0) {
      createWindow()
    }
  })
})
})
```



ipcMain, ipcRenderer

```
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

let window;
ipcMain.on('nazwaKanału', (e, args) => {
  console.log(args);
});
```

ipcMain, ipcRenderer

ipcMain, ipcRenderer

obiekty umożliwiające komunikację pomiędzy procesem głównym a przeglądarką

Wybrane elementy	Opis
<code>on(channel, listener)</code>	metoda nasłuchuje na kanale (kanał określony przez dowolny String), po przechwyceniu wiadomości na kanał wywołany jest kod metody listener
<code>once(channel, listener)</code>	- - uruchamiany tylko raz
<code>removeListener(channel, listener)</code>	usuwa słuchacza zdarzeń
<code>send(channel, data)</code>	wysyła dane na określony kanał
<code>sendSync</code>	wymusza synchroniczne wysyłanie wiadomości (blokuje proces)
<code>handle (Main proces)</code>	uruchamia funkcję zwracając wynik do procesu Renderera
<code>invoke (Renderer proces)</code>	zdalne wywołanie funkcji z procesu głównego po stronie Renderera

ipcMain, ipcRenderer

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1 id="header"> Okno 1</h1>
  <button id="btn">Wyślij wiadomość</button>
  <script>
    const { ipcRenderer } = require('electron');

    let button = document.querySelector("#btn");
    button.addEventListener("click", (e) => {
      ipcRenderer.send('nazwaKanału', 'wiadomość dla procesu głównego '+Math.random());
    });
  </script>
</body>
</html>
```

ipcMain, ipcRenderer

```

<!DOCTYPE html>
<html>
  <body>
    <button>
  </body>
</html>

```

```

MINGW64/e/electron1
root@DESKTOP-CJUPR7C MINGW64 /e/electron1
$ npm run start
> electron1@1.0.0 start E:\electron1
> electron .

(node:11436) electron: The default of nativeWindowOpen is deprecated and will be
changing from false to true in Electron 15. See https://github.com/electron/el
ectron/issues/28511 for more information.
(Use 'electron --trace-warnings ...' to show where the warning was created)

```

Okno 1

Wyślij wiadomość

ipcMain, ipcRenderer

```

<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1 id="header"> Okno 1</h1>
  <button id="btn">Wyślij wiadomość</button>
  <script>
    const {ipcRenderer} = require('electron');

    let button = document.querySelector("#btn");
    button.addEventListener("click", (e)=>{
      ipcRenderer.send('nazwaKanal', 'wiadomość dla procesu głównego '+Math.random());
    })

    ipcRenderer.on('nazwaKanal', (e, args)=>{
      const header = document.querySelector("#header");
      header.innerHTML = args;
    })
  </script>

```

ipcMain, ipcRenderer

```

const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

let window;
ipcMain.on('nazwaKanal', (e, args)=>{
  console.log(args);
  e.sender.send('nazwaKanal', 'Wiadomość odebrana'+Math.random());
});

```

ipcMain, ipcRenderer

```

<!DOCTYPE html>
<html>
  <body>
    <button>
  </body>
</html>

```

```

MINGW64/e/electron1
root@DESKTOP-CJUPR7C MINGW64 /e/electron1
$ npm run start
> electron1@1.0.0 start E:\electron1
> electron .

(node:10012) electron: The default of nativeWindowOpen is deprecated and will be
changing from false to true in Electron 15. See https://github.com/electron/el
ectron/issues/28511 for more information.
(Use 'electron --trace-warnings ...' to show where the warning was created)

```

Okno 1

Wyślij wiadomość

```

wiadomość dla procesu głównego 0.5928049198694705
wiadomość dla procesu głównego 0.03798861167465062
wiadomość dla procesu głównego 0.5909035248891152
wiadomość dla procesu głównego 0.2594533993818531
wiadomość dla procesu głównego 0.6952469818445755
wiadomość dla procesu głównego 0.4645438815862064
wiadomość dla procesu głównego 0.8960484426731914
wiadomość dla procesu głównego 0.9629861586895208
wiadomość dla procesu głównego 0.5747166279570894
wiadomość dla procesu głównego 0.44448388574416997
wiadomość dla procesu głównego 0.4694501882268116

```

```

const header = document.querySelector("#header");
header.innerHTML = args;

```

Wysyłanie wiadomości do konkretnego okna:

```
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

let window;

function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', () => { window = null; });
  window.webContents.on('did-finish-load', e => {
    setInterval(() => {
      window.webContents.send('nazwaKanału', 'Wiadomość: ' + Math.random());
    }, 500);
  });
}
```

ipcMain, ipcRenderer

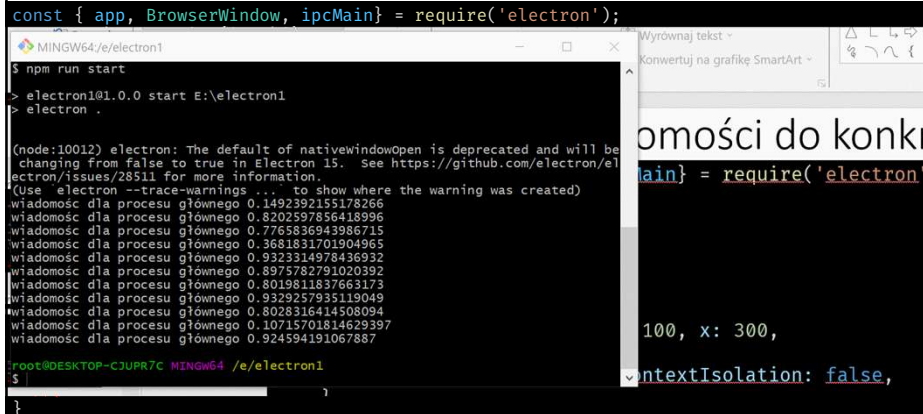
```
const { app, BrowserWindow, ipcMain, dialog } = require('electron');
const path = require('path');

let window;

async function selectLanguage() {
  let languages = ['Java', 'C++', 'JavaScript'];
  let result = await dialog.showMessageBox({
    message: 'Jaki język programowania preferujesz?',
    buttons: languages,
  });
  return languages[result.response];
}

ipcMain.handle('dialog', e => {
  return selectLanguage();
})
```

Wysyłanie wiadomości do konkretnego okna:



```
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');

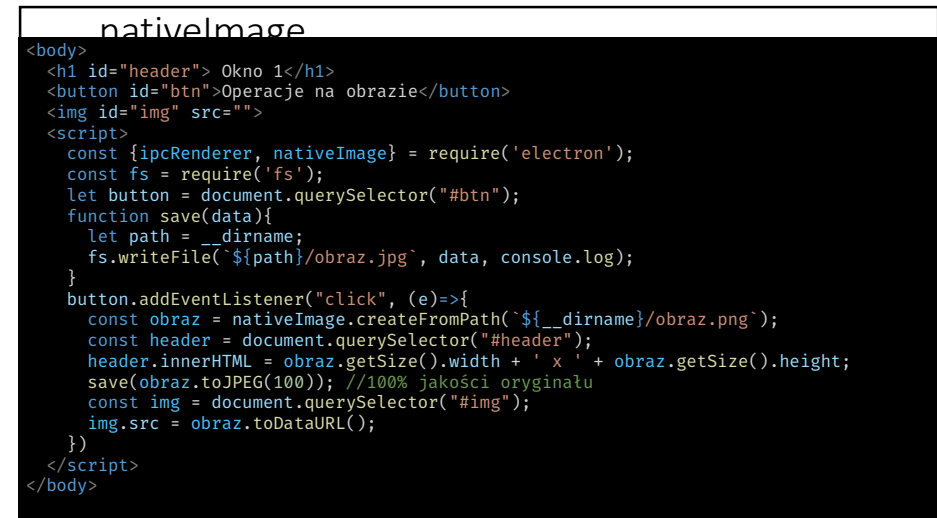
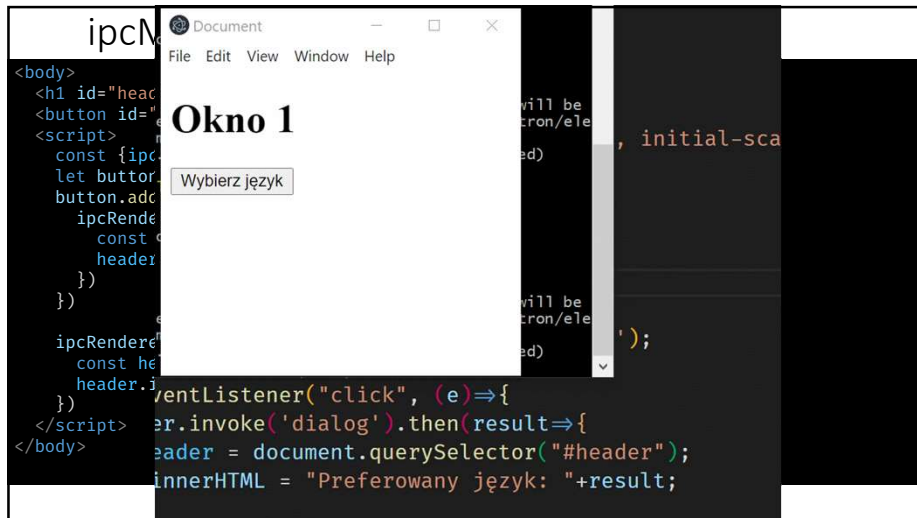
let window;

function createWindow() {
  window = new BrowserWindow({
    width: 300, height: 300, y: 100, x: 300,
    webPreferences: {
      nodeIntegration: true, contextIsolation: false,
    },
  });
  window.loadFile('index.html');
  window.on('closed', () => { window = null; });
  window.webContents.on('did-finish-load', e => {
    setInterval(() => {
      window.webContents.send('nazwaKanału', 'Wiadomość: ' + Math.random());
    }, 500);
  });
}
```

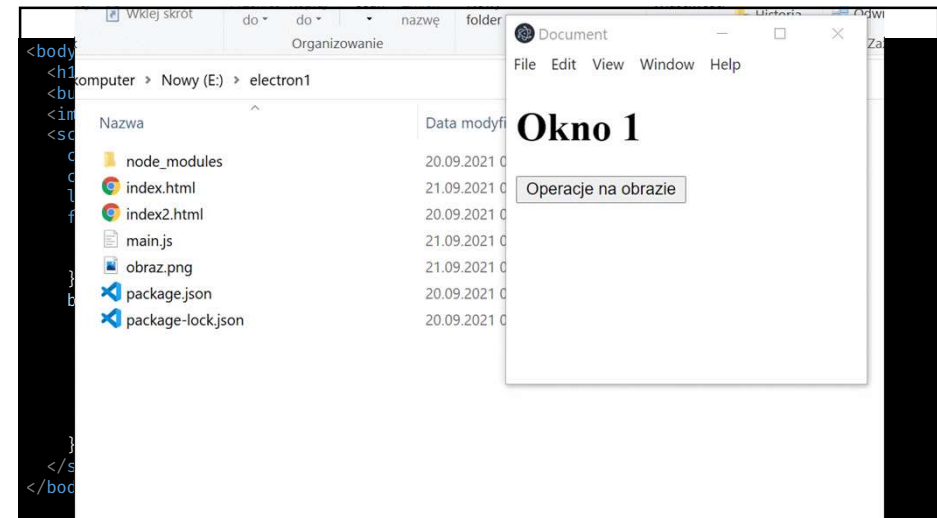
ipcMain, ipcRenderer

```
<body>
<h1 id="header"> Okno 1</h1>
<button id="btn">Wybierz język</button>
<script>
  const { ipcRenderer, remote } = require('electron');
  let button = document.querySelector("#btn");
  button.addEventListener("click", (e) => {
    ipcRenderer.invoke('dialog').then(result => {
      const header = document.querySelector("#header");
      header.innerHTML = "Preferowany język: " + result;
    })
  })

  ipcRenderer.on('nazwaKanału', (e, args) => {
    const header = document.querySelector("#header");
    header.innerHTML = args;
  })
</script>
</body>
```



nativeImage	
nativeImage	
umożliwia wykonywanie operacji na plikach graficznych (png, jpg)	
Wybrane elementy	Opis
<code>createEmpty()</code>	tworzy pusty obiekt NativeImage
<code>createFromPath(path)</code>	tworzy obiekt z pliku
<code>createFromBitmap(buffer, options)</code>	tworzy obiekt z bufora zawierającego dane pikseli
<code>createFromBuffer(buffer[, options])</code>	tworzy obiekt z bufora (próbuję odkodować wpierw png, później jpg)
<code>createFromDataURL(dataURL)</code>	- - dataURL
<code>toPNG, toJPEG, toBitmap, toDataURL</code>	zwraca skonwertowany obiekt w postaci bufora/tekstu
<code>crop(rect), resize(options)</code>	operacje graficzne (przycięcie, zmiana rozmiaru)



Używanie modułów natywnych

W przypadku modułów natywnych, należy je dostosować do zainstalowanej wersji electron.

W tym celu moduł należy przebudować:

instalacja narzędzia:

```
npm install -g electron-rebuild
```

przebudowanie:

```
electron-rebuild nazwa-modułu
```

Tworzenie aplikacji

```
npm install --save-dev electron-builder
```

w package.json dodać do sekcji "scripts":

```
"scripts": {
  "start": "electron .",
  "build": "electron-builder -w"
},
```

```
root@DESKTOP-CJUPR7C MINGW64 /d/electron_app
$ npm run build
> electron_app@1.0.0 build D:\electron_app
> electron-builder -w
• electron-builder version=22.10.5 os=10.0.19041
• description is missed in the package.json appPackageFile=D:\electr
ckage.json
• packaging platform=win32 arch=x64 electron=12.0.7 appOutDir=d
npacked
```

Tworzenie aplikacji

```
npm install --save-dev electron-builder
```

w package.json dodać do sekcji "scripts":

```
"scripts": {
  "start": "electron .",
  "build": "electron-builder -w"
},
```

Tworzenie aplikacji

Organizowanie				
Ten komputer > Nowy (D:) > electron_app > dist				
Nazwa	Data modyfikacji	Typ	Rozmiar	
win-unpacked	28.05.2021 17:50	Folder plików		
builder-debug.yml	28.05.2021 17:50	Plik YML	6 KB	
electron_app Setup 1.0.0.exe	28.05.2021 17:50	Aplikacja	57 311 KB	
electron_app Setup 1.0.0.exe.blockmap	28.05.2021 17:50	Plik BLOCKMAP	61 KB	

Dostęp do procesu możliwości procesu main

```
const { app, BrowserWindow, dialog, ipcMain } = require('electron')  
...  

```

```
ipcMain.handle('dialog', async (e) => {  
  const wynik = await dialog.showOpenDialog(window, {  
    buttonLabel: "Wybierz plik",  
    defaultPath: app.getPath('desktop'),  
  })  
  return wynik;  
});
```

```
const { ipcRenderer } = require('electron');  
async function x() {  
  const wynik = await ipcRenderer.invoke('dialog');  
  p.innerHTML = wynik.filePaths[0];  
}
```