Extensiones y consejos Visual Studio Code

Debugger for java	Microsoft
Dev Containers	Microsoft
Docker	Microsoft
Error Lens	Alexander
Extension pack for java	Microsoft
Git Graph	Mhutchie
Gradle for Java	Microsoft
intelliCode	Microsoft
intelliCode API Usage Examples	Microsoft
JavaScrpt (ES6) code snippets	Charalampos karypidis
Language Support for Java(TM) by Red Hat	Red Hat
Live Server	Ritwick Dey
Project Manager for Java	Microsoft
Pylance	Microsoft
Python	Microsoft
Python Debugger	Microsoft
Rainbow Brackets	Mhammed Talhouy
Test Runner for Java	Microsoft
Color Picker Universal	Jeronimo Ekerdt
Image preview	Kiss Tamás

Ctrl + ,
Wrap Lines en **on** para que haga salto de línea “visual”

- Styles en misma línea con Join (Shortcut)
Ejemplo de resultado con Join:

```
style: {
        display: "flex", flexWrap: "wrap", justifyContent: "space-between", gap:
"20px",
      }
```

- Si no lo pongo en ejemplos es para facilitaros la comprensión, pero luego toca hacer Join
- No hace falta TODO el style en la misma línea, puedes agrupar según necesidad. En el siguiente ejemplo he agrupado la parte relacionada con “flex”

```
style: {
        display: "flex", flexWrap: "wrap",
        justifyContent: "space-between", gap: "20px",
      }
```

- Para el nombre de las funciones y variables usaremos **camelCase,** para las constantes **UPPER_CASE**

Estructura simple de un componente en Mithril

```
function texto() {
 return {
   view: function () {
     return m("div", {
       style: {
         margin: "20px 0px"
       }
     },
       m("h1", {
         style: {
           margin: "0",
           marginBottom: "15px"

         }
       }, "Ejemplo Texto"),

       m("p", {
         style: {
           margin: "0",
           border: "1px solid black",
           padding: "6px"
         }
       }, "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Proin imperdiet eros sed volutpat vulputate.")
     );
   }
 };
}
```

Importar/Exportar la página

Archivo **index.js**

```
import { inicio, otraRuta} from "./components.js"

const routes = {
   "/Inicio": { view: () =>  m(inicio)},
   "/OtraRuta": { view: () =>  m(otraRuta)},
}

m.route(document.body, "/Inicio", routes)
```

Para que esto vaya, hay que exportarlo al final del archivo **components.js**

```
export { inicio, otraRuta };
```

Cada elemento en export será una “página”, luego Inicio llamará a los distintos componentes como Header o Main, estos pueden estar llamando a otros componentes (functions)

```
function inicio() {
 return {
   oncreate: () => {
     window.scrollTo(0, 0); //Esto es para que vaya arriba la ventana
   },
   view: () => [
     m("div", {
       style: {/*backgroundColor: "grey"*/},
     },
       m(header),
       m(blancoHeader),
       m("div", {
         style: {
           display: "flex", flexDirection: "row", width: "100%",
boxSizing: "border-box", justifyContent: "center", gap: "32px",
padding: "0px 10%", margin: "0 auto",
         },
       },
         //se puede poner el contenido del main directamente
         m(main),
         //m(aside),
       ),
       //m(blanco),
       m(footer),
     )
   ]
 }
}
```

-Le he añadido estilo para que todo lo que coloques en m(main) tenga un padding, m(aside) también lo tendría aunque no se use.
-m(blanco) es un componente que solamente es un div con espacio para ocupar y que se vea el footer al final

```
function blanco() {
 return {
   view: ({ }) => [
     m("div",
       {
         style: {
           width: "100%", height: "900px", "box-sizing": "border-box",
         },
       },
     ), ""
   ],
 };
}
```

Ejemplos

Cuestionario/input

```
let nombreCuestionario = "";

m("input", {
     type: "text",
     placeholder: "Nombre",
     value: nombreCuestionario,
     onchange: (e) => { nombreCuestionario = e.target.value; },
     style: inputEstilos,
   }),
```

Imagen

Intentad usar .webp para las imágenes
https://cloudconvert.com/jpeg-to-webp

```
m("img", { src: "/img/MainLogo.png", alt: "Logo", style: { height:
"40px" } }),
```

Link

```
m("a", { href: "#", style: linkStyle }, "Página 1"),
```

Model

```
function Posts() {
 return {
   select: {
     model: {
       tarjetas: [
         {
           titulo: "Post 1",
           imagen: "/img/descarga.jpeg",
           descripcion: "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Proin imperdiet eros sed volutpat vulputate. Aenean
varius quis massa ac vestibulum. Donec non diam libero. Curabitur vitae
dignissim arcu."
         },
         {
           titulo: "Título 2",
           imagen: "/img/Simpsons.jpg",
           descripcion: "Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Proin imperdiet eros sed volutpat vulputate. Aenean
varius quis massa ac vestibulum. Donec non diam libero. Curabitur vitae
dignissim arcu."
         },
         {
           titulo: "Título 3",
           imagen: "https://via.placeholder.com/150",
           descripcion: "Descripción de la tarjeta 3."
         },
         {
           titulo: "Título 4",
           imagen: "https://via.placeholder.com/150",
           descripcion: "Descripción de la tarjeta 4."
         },
         {
           titulo: "Título 5",
           imagen: "https://via.placeholder.com/150",
           descripcion: "Descripción de la tarjeta 5."
         },
         {
           titulo: "Título 6",
           imagen: "https://via.placeholder.com/150",
           descripcion: "Descripción de la tarjeta 6."
         },
```

```
      ]
    },
  },

  view: function () {
    return m("div", {
      style: {
        display: "flex",
        flexWrap: "wrap",
        justifyContent: "space-between",
        gap: "20px"
      },
    },
      m("h1", {
        style: {
          textAlign: "left",
          width: "100%",
          margin: "0"
        }
      }, "Posts"),

      // Renderizar las tarjetas desde el modelo
      this.select.model.tarjetas.map((tarjeta) =>
        m("div", {
          style: {
            width: "400px",
            height: "500px",
            backgroundColor: "green",
            display: "flex",
            flexDirection: "column",
            justifyContent: "space-between",
            alignItems: "center",
            borderRadius: "12px",
            boxSizing: "border-box",
            boxShadow: "0px 4px 12px rgba(0, 0, 0, 0.8)",
            overflow: "hidden"
          },
        },
          m("h3", {
            style: {
              margin: "0",
              height: "50px",
              width: "100%",
```

```
            textAlign: "center",
            backgroundColor: "blue",
            display: "flex",
            justifyContent: "center",
            alignItems: "center"
          },
        }, tarjeta.titulo),

        m("div", {
          style: {
            width: "100%",
            height: "70%",
            overflow: "hidden",
            display: "flex",
            justifyContent: "center",
            alignItems: "center"
          },
        },
          m("img", {
            src: tarjeta.imagen,
            alt: `Imagen de ${tarjeta.titulo}`,
            style: {
              width: "100%",
              height: "100%",
              objectFit: "cover"
            }
          })
        ),

        m("p", {
          style: {
            margin: "0",
            padding: "10px",
            textAlign: "center",
            height: "150px",
            width: "100%",
            boxSizing: "border-box",
            backgroundColor: "yellow",
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
            overflow: "hidden"
          },
```

```
          }, tarjeta.descripcion)
        )
      )
    );
  },
};
}
```

Model con Index (Tarjetas seleccionables) mostrando la tarjeta seleccionada

```
function posts() {
 return {
   select: {
     model: {
       tarjetas: [
         { titulo: "Post 1", imagen: "/img/descarga.jpeg", descripcion:
"Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin
imperdiet eros sed volutpat vulputate. Aenean varius quis massa ac
vestibulum. Donec non diam libero. Curabitur vitae dignissim arcu." },
         { titulo: "Título 2", imagen: "/img/Simpsons.jpg",
descripcion: "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Proin imperdiet eros sed volutpat vulputate. Aenean varius quis massa
ac vestibulum. Donec non diam libero. Curabitur vitae dignissim arcu."
},
         { titulo: "Título 3", imagen:
"https://via.placeholder.com/150", descripcion: "Descripción de la
tarjeta 3." },
         { titulo: "Título 4", imagen:
"https://via.placeholder.com/150", descripcion: "Descripción de la
tarjeta 4." },
         { titulo: "Título 5", imagen:
"https://via.placeholder.com/150", descripcion: "Descripción de la
tarjeta 5." },
         { titulo: "Título 6", imagen:
"https://via.placeholder.com/150", descripcion: "Descripción de la
tarjeta 6." }
       ],
     selectedPost: null, // Al principio no hay ninguna seleccionada
   },
 },

   selectPost: function (index) {
```

```
    this.select.model.selectedPost = index;
  },

  view: function () {
    return m("div", {
      style: {
        display: "flex", flexDirection: "column", gap: "20px"
      }
    },
      m("h1", { style: { textAlign: "left", margin: "0" } }, "Posts"),

      // Contenedor de las tarjetas
      m("div", {
        style: {
          display: "flex", flexWrap: "wrap", justifyContent:
"space-between", gap: "20px",
        }
      },
        this.select.model.tarjetas.map((tarjeta, index) =>
          m("div", {
            onclick: () => this.selectPost(index),
            style: {
              width: "400px", height: "500px", display: "flex",
flexDirection: "column", justifyContent: "space-between", alignItems:
"center", borderRadius: "12px", boxSizing: "border-box", boxShadow:
"0px 4px 12px rgba(0, 0, 0, 0.8)", overflow: "hidden",
              outline: this.select.model.selectedPost === index ? "4px
solid cyan" : "none",
            },
          },
            m("h3", { style: { margin: "0", height: "50px", width:
"100%", textAlign: "center", backgroundColor: "blue", display: "flex",
justifyContent: "center", alignItems: "center" } }, tarjeta.titulo),
            m("div", { style: { width: "100%", height: "70%",
overflow: "hidden", display: "flex", justifyContent: "center",
alignItems: "center" } },
              m("img", { src: tarjeta.imagen, alt: `Imagen de
${tarjeta.titulo}`, style: { width: "100%", height: "100%", objectFit:
"cover" } })
            ),
            m("p", { style: { margin: "0", padding: "10px", textAlign:
"center", height: "150px", width: "100%", boxSizing: "border-box",
```

```
backgroundColor: "yellow", display: "flex", justifyContent: "center",
alignItems: "center", overflow: "hidden" } }, tarjeta.descripcion)
            )
          )
        ),

        // Bloque de tarjeta seleccionada
        m("div", {
          style: {
            width: "100%",
            marginTop: "20px"
          }
        },
          m("h1", { style: { margin: "0", textAlign: "left" } },
"Tarjeta seleccionada:"),
          m("p", { style: { margin: "5px 0",  } },
            this.select.model.selectedPost !== null
              ?
this.select.model.tarjetas[this.select.model.selectedPost].titulo
              : "Ninguna tarjeta seleccionada"
          )
        ),
      );
    },
  };
}
```

Cambiar de página

- Además de añadir lo del código, habrá que exportarlo e importarlo de forma correcta

```
m(m.route.Link, {
          href: "/Pagina2",
          style: linkStyle
        }, "Página 2"),
```

Persistencia de datos

- La guardamos en localStorage, lo que hace que se guarde en el propio navegador, las cookies, vaya.

```
// Componente para mostrar el número y los botones
function contador() {
 return {
   oninit: function() {
```

```
    // Intentamos obtener el número guardado en el localStorage, si no
existe, lo inicializamos a 10
    this.numero = parseInt(localStorage.getItem("numero") || "10");
  },

  // Función para actualizar el valor en el localStorage
  actualizarStorage: function() {
    localStorage.setItem("numero", this.numero);
  },

  sumar: function() {
    this.numero += 1;
    this.actualizarStorage();
  },

  restar: function() {
    this.numero -= 1;
    this.actualizarStorage();
  },

  view: function() {
    return m("div", {
      style: {
        textAlign: "center",
        marginTop: "20px"
      }
    },
      m("h1", {
        style: {
          margin: "0",
          marginBottom: "15px"
        }
      }, "Persistencia de datos"),
      m("h2", this.numero), // Mostrar el número
      m("div", {
        style: {
          marginTop: "10px",
          display: "flex",
          justifyContent: "center",
          gap: "20px"
        }
      },
        m("button", {
```

```
            style: {
              padding: "10px",
              fontSize: "16px",
              cursor: "pointer"
            },
            onclick: () => this.restar() // Resta 1
          }, "-"),

          m("button", {
            style: {
              padding: "10px",
              fontSize: "16px",
              cursor: "pointer"
            },
            onclick: () => this.sumar() // Suma 1
          }, "+")
        )
      );
    }
  };
}
```

Fetch

- Se usa para hacer llamadas a una base de datos, ya sea una API o un .json.
- Para el ejemplo usa una API donde hay muchos posts generados por varios usuarios, en este caso se mostrará el título y el cuerpo de cada post del usuario indicado, en este caso el 1.
- Para hacer pruebas podeis usar https://jsonplaceholder.typicode.com/ o cualquier api gratuita.

```
let usuarioFetch = 1;

function pruebaFetch() {
 return {
   posts: [],

   oninit: function() {
     fetch("https://jsonplaceholder.typicode.com/posts")  // Puedes
apuntar a un archivo local .json
       .then(response => response.json())
       .then(data => {
         this.posts = data.filter(post => post.userId ===
usuarioFetch);
         m.redraw();
```

```
      })
      .catch(error => {
        console.error("Error al cargar los posts:", error);
      });
  },

  view: function() {
    return m("div",
      m("h1", `Listado de Posts (userId = ${usuarioFetch})`),

      m("ul", {
        style: {
          listStyleType: "none",  // Sin puntos
          padding: 0,
          margin: 0,
        }
      },
        this.posts.map(post =>
          m("li", {
            key: post.id,
            style: {
              marginBottom: "20px",
              padding: "10px",
              border: "1px solid #ddd",
              borderRadius: "8px",
              backgroundColor: "#f9f9f9"
            }
          },
            m("h3", {
              style: { margin: "0", display: "block", marginBottom:
"5px",}
            }, post.title),

            m("p", {
              style: { margin: 0, color: "#555", fontSize: "14px" }
            }, post.body)
          )
        )
      )
    );
  }
};
}
```

Tarjetas / posts usando model con .json

Hablando con Pablo me dijo que es mejor usar this.select.model o this.model puesto que es más genérico

```
function posts() {
 return {
   select: {
     model: {
       tarjetas: [],  // Inicialmente vacío, se llenará con el JSON
       selectedPost: null,
     },
   },

   oninit: function () {
     m.request({
       method: "GET",
       url: "./bases/postspracticas.json"  // Ruta relativa dentro de
tu proyecto
     }).then((data) => {
       this.select.model.tarjetas = data.posts;  // data.posts es el
array del JSON
     });
   },

   selectPost: function (index) {
     this.select.model.selectedPost = index;
   },

   view: function () {
     return m("div", {
       style: {
         display: "flex", flexDirection: "column", gap: "20px"
       }
     },
       m("h1", { style: { textAlign: "left", margin: "0" } }, "Posts"),

       // Contenedor de las tarjetas
       m("div", {
         style: {
           display: "flex", flexWrap: "wrap", justifyContent:
"space-between", gap: "40px",
         }
       },
         this.select.model.tarjetas.map((tarjeta, index) =>
```

```
          m("div", {
            onclick: () => this.selectPost(index),
            style: {
              width: "400px", height: "500px", display: "flex",
flexDirection: "column", justifyContent: "space-between", alignItems:
"center", borderRadius: "12px", boxSizing: "border-box", boxShadow:
"0px 4px 12px rgba(0, 0, 0, 0.8)", overflow: "hidden",
              outline: this.select.model.selectedPost === index ? "4px
solid cyan" : "none",
            },
          },
            m("h3", { style: { margin: "0", height: "50px", width:
"100%", textAlign: "center", backgroundColor: "white", display: "flex",
justifyContent: "center", alignItems: "center" } }, tarjeta.titulo),
            m("div", { style: { width: "100%", height: "70%",
overflow: "hidden", display: "flex", justifyContent: "center",
alignItems: "center" } },
              m("img", { src: tarjeta.imagen, alt: `Imagen de
${tarjeta.titulo}`, style: { width: "100%", height: "100%", objectFit:
"cover" } })
            ),
            m("p", { style: { margin: "0", padding: "10px", textAlign:
"center", height: "150px", width: "100%", boxSizing: "border-box",
backgroundColor: "white", display: "flex", justifyContent: "center",
alignItems: "center", overflow: "hidden" } }, tarjeta.descripcion)
          )
        )
      ),

      // Bloque de tarjeta seleccionada
      m("div", {
        style: {
          width: "100%",
          marginTop: "20px"
        }
      },
        m("h1", { style: { margin: "0", textAlign: "left" } },
"Tarjeta seleccionada:"),
        m("p", { style: { margin: "5px 0" } },
          this.select.model.selectedPost !== null
            ?
this.select.model.tarjetas[this.select.model.selectedPost].titulo
            : "Ninguna tarjeta seleccionada"
```

```
          )
        ),
      );
    },
  };
}
```

Header / menú dinámico

Header que llama a los dos tipos de header

```
function header() {
 let width = window.innerWidth;

 window.addEventListener("resize", () => {
   width = window.innerWidth;
   m.redraw();
 });

 return {
   view: function () {
     return width < 768 ? m(HeaderBurger) : m(HeaderNormal);
   },
 };
}
```

Header hamburguesa (desplegado)

```
function NavBurger() {
 let styleLi = {
   listStyle: "none",
   display: "flex",
   alignItems: "center",
   justifyContent: " center",
   width: "30%",
   borderBottom: "1px solid white",
   textAlign: "center",
```

```
};

let show = false;

return {
  view: () => {
    return m(
      "nav",
      {
        style: {
          display: "flex",
          alignItems: "center",
          justifyContent: "end",
          height: alturaHeader,
          width: "100%",
          position: "relative",
        },
      },
      m("i.fa-solid.fa-bars", {
        onclick: () => {
          show = !show;
          m.redraw();
        },
        style: { color: "white", fontSize: "30px", cursor:
"pointer" },
      }),
      show &&
        m(
          "ul",
          {
            style: {
              display: "flex",
              flexDirection: "column",
              alignItems: "center",
```

```
            justifyContent: "center",
            gap: "30px",
            listStyle: "none",
            background: "#466874",
            position: "absolute",
            top: alturaHeader,
            padding: "20px",
            width: "145%",
            margin: "0",
            border: "1px solid white",
            right: "-66px",
          },
        },
```

Conversor de html/css (el html tiene que tener el CSS) a Mithril

https://arthurclemens.github.io/mithril-template-converter/index.html

Ciclo de Vida de Mithril

Fases principales

Método	Cuándo se ejecuta	Uso típico
`oninit`	Antes de que el componente se monte en el DOM	Inicialización de datos
`oncreate`	Justo después de que el DOM se ha creado por primera vez	Configurar animaciones, eventos, manipular el DOM
`onupdate`	Cuando se actualiza el DOM tras un `m.redraw()`	Reaccionar a cambios de datos, actualizar UI
`onbeforeremove`	Antes de que un nodo sea eliminado del DOM	Agregar animaciones de salida

`onremove`	Después de que el nodo se ha eliminado	Limpiar eventos o recursos

Nueva forma de usar flex

flexGrow: “1” -> Que no crezca
flexShrink: “0” -> Para que no se haga pequeña
flexBasis: “300px” -> Tamaño base

todo eso equivale a:
flex: “1 0 300px”

Yo he utilizado esto para adecuarlo a mis necesidades:

```
display: "flex",
flex: "1 0 200px",
maxHeight: "300px",
maxWidth: "450px",
```