Contents

Template Bindings

```
export default function App() {
  const name = "John Wick";
  const count = 1;
  const toggled = true;
  const html = "<strong>Highlighted</strong>";

  return (
    <>
      <p>Hello {name}</p>
      <p>{`Hello ${name}`}</p>
      <p>Count: {count}, Doubled {count * 2}</p>
      <p>Toggled: {toggled ? "  " : "  "}</p>
      <p innerHTML="<em>Emphasized</em>"></p>
      <p innerHTML={html}></p>
    </>
  );
}
```

Reactivity

```
import { createSignal } from "solid-js";

export default function App() {
  const [count, setCount] = createSignal(0);
  const [numbers, setNumbers] = createSignal([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);
  const [profile, setProfile] = createSignal({ name: "John Wick", active: true });

  return (
    <>
      {/* Update primitive type */}
      <button onClick={() => setCount((count) => count + 1)}>Count: {count()}</button>

      {/* Update array */}
      <button
        onClick={() =>
          setNumbers((numbers) => [...numbers, numbers.length + 1])
        }
      >
        {JSON.stringify(numbers())}
      </button>

      {/* Update object */}
      <button
        onClick={() =>
          setProfile((profile) => ({ ...profile, active: !profile.active }))
        }
      >
        {JSON.stringify(profile())}
      </button>
    </>
  );
}
```

Computed/Derived Values

```
import { createMemo, createSignal } from "solid-js";

export default function App() {
  const [count, setCount] = createSignal(0);
  const doubled = count() * 2;
  // ...or cache results if doing expensive computations:
  // const doubled = createMemo(() => count() * 2);

  return (
     <button onClick={() => setCount((count) => count + 1)}>
       Count: {count} Doubled: {doubled}
     </button>
  );
}
```

Loops

```
import { createSignal, Index, For } from "solid-js";

export default function App() {
  const [todos, setTodos] = createSignal([
    { id: 1, title: "Clean room" },
    { id: 2, title: "Wash dishes" },
    { id: 3, title: "Buy milk" }
  ])

  return (
    <ul>
      {/* This works similar to React (non-keyed) */}
      {todos().map((todo, index) => (
        <li>
          {index}: {todo.title}
        </li>
      ))}

      {/* ..but this is the recommended method (keyed by reference) */}
      <For each={todos()}>
        {(todo, index) =>
          <li>
            {index()}: {todo.title}
          </li>
        }
      </For>

      {/* ..or this (keyed by index) */}
      <Index each={todos()}>
        {(todo, index) =>
          <li>
            {index}: {todo().title}
          </li>
        }
      </Index>
    </ul>
  );
}
```

Conditional Rendering

```
import { createSignal, Show } from "solid-js";
import { createSignal, Match, Show, Switch } from "solid-js";

export default function App() {
  const [toggled, setToggled] = createSignal(false);
  const [count, setCount] = createSignal(0);

  return (
    <>
      <button onClick={() => setToggled((toggled) => !toggled)}>
        Toggle
      </button>

      {/* This works similar to React */}
      {toggled && <p>Visible</p>}f
      {toggled() ? <p>Toggle: On</p> : <p>Toggle: Off</p>}

      {/* ..but this is the recommended method */}
      <Show when={toggled()}><p>Visible</p></Show>
      <Show when={toggled()} fallback={<p>Toggle: Off</p>}><p>Toggle: On</p></Show>

      <button onClick={() => setCount(count => count + 1)}>
        Count: {count()}
      </button>

      {/* For multiple mutually exclusive conditions */}
      <Switch fallback={<p>Not divisible by either 2 or 3</p>}>
        <Match when={count() % 2 === 0}>
          <p>Divisible by 2</p>
        </Match>
        <Match when={count() % 3 === 0}>
          <p>Divisible by 3</p>
        </Match>
      </Switch>
    </>
  );
}
```

Component Props

```
import { mergeProps } from "solid-js";

const BookCard = (props) => {
  props = mergeProps({ title: "Unknown", author: "Unknown"}, props);

  return (
    <div>
      <p>Title: {props.title}</p>
      <p>Author: {props.author}</p>
    </div>
  )
}

const ProfileCard = (props) => {
  return (
    <div>
      <p>Name: {props.profile.name}</p>
      <p>Location: {props.profile.location}</p>
    </div>
  );
}

export default function App() {
  const book = {
    title: "John Wick #2",
    author: "Greg Pak"
  };
  const profile = {
    name: "John Wick",
    location: "USA"
  };

  return (
    <>
      {/* Use default values for props */}
      <BookCard />
      {/* Use static values for props */}
      <BookCard title="John Wick #1" author="Greg Pak" />
      {/* Use dynamic values for props */}
      <BookCard title={book.title} author={book.author} />
      <ProfileCard profile={profile} />
    </>
  );
}
```

Component Slots

```
import { Show } from "solid-js";

const Layout = (props) => {
  return (
    <div>
      <div>{props.header}</div>
      <Show when={props.sidebar}>{props.sidebar}</Show>
      <div>{props.children}</div>
      <Show when={props.footer} fallback={<p>Powered by SolidJS</p>}>{props.footer}</Show>
    </div>
  );
};

export default function App() {
  return (
    <Layout header={<h1>Title</h1>}>
      Lorem ipsum dolor sit amet...
    </Layout>
  );
}
```

Component Lifecycle

```
import { createSignal, onMount, onCleanup, Show } from "solid-js";

const Message = () => {
  onMount(() => console.log("Mounted"));

  onCleanup(() => console.log("Unmounted"));

  return <p>Hello</p>;
};

export default function App() {
  const [toggled, setToggled] = createSignal(false);

  return (
    <>
      <button onClick={() => setToggled((toggled) => !toggled)}>Toggle</button>
      <Show when={toggled()}><Message /></Show>
    </>
  );
}
```

DOM Events

```
import { createSignal } from "solid-js";

export default function App() {
  const [count, setCount] = createSignal(0);

  const increment = (e) => setCount((count) => count + 1);

  return (
    <>
      {/* Handle event inline */}
      <button onClick={() => setCount((count) => count + 1)}>Count: {count()}</button>
      {/* Handle event in function */}
      <button onClick={increment}>Count: {count}</button>
    </>
  );
}
```

Custom Events

```
import { createSignal } from "solid-js";

const Counter = (props) => {
  const [count, setCount] = createSignal(0);

  const increment = () => {
    setCount((count) => count + 1);

    // Not really a custom event, more of a callback.
    props.onIncremented && props.onIncremented({ count });
  };

  return <button onClick={increment}>Count: {count}</button>;
};

export default function App() {
   const [count, setCount] = createSignal(0);

  const handleIncremented = (e) => {
    setCount(e.count);
  };

  return (
    <>
      <p>Count: {count()}</p>
      <Counter onIncremented={handleIncremented} />
    </>
  );
}
```

Form Input Bindings

```
import { createSignal } from "solid-js";

export default function App() {
  const [name, setName] = createSignal("John Wick");
  const [isActive, setIsActive] = createSignal(true);
  const [gender, setGender] = createSignal("M");
  const [rating, setRating] = createSignal(5);
  const [notes, setNotes] = createSignal("Lorem ipsum dolor sit amet");

  return (
    <form>
      <div>
        <label>
          Name
          <input
            type="text"
            value={name()}
            onInput={(e) => setName(e.target.value)}
          />
        </label>
      </div>
      <div>
        <label>
          Active
          <input
            type="checkbox"
            checked={isActive()}
            onChange={(e) => setIsActive(e.target.checked)}
          />
        </label>
      </div>
      <div>
        <label>
          Gender
          <select
            value={gender()}
            onChange={(e) => setGender(e.target.value)}
          >
            <option value="M">Male</option>
            <option value="F">Female</option>
          </select>
        </label>
      </div>
      <div>
        Level
        <label>
          <input
            type="radio"
            name="rating"
            value={1}
            checked={rating() === 1}
            onChange={(e) => setRating(+e.target.value)}
          />{" "}
          1
        </label>
        <label>
          <input
```

```
            type="radio"
            name="rating"
            value={2}
            checked={rating() === 2}
            onChange={(e) => setRating(+e.target.value)}
          />{" "}
          2
        </label>
        <label>
          <input
            type="radio"
            name="rating"
            value={3}
            checked={rating() === 3}
            onChange={(e) => setRating(+e.target.value)}
          />{" "}
          3
        </label>
        <label>
          <input
            type="radio"
            name="rating"
            value={4}
            checked={rating() === 4}
            onChange={(e) => setRating(+e.target.value)}
          />{" "}
          4
        </label>
        <label>
          <input
            type="radio"
            name="rating"
            value={5}
            checked={rating() === 5}
            onChange={(e) => setRating(+e.target.value)}
          />{" "}
          5
        </label>
      </div>
      <div>
        <label>
          Notes
          <textarea
            value={notes()}
            onInput={(e) => setNotes(e.target.value)}
          />
        </label>
      </div>
    </form>
  );
}
```

State Management

```
import { render } from "solid-js/web";
import { createStore, produce } from "solid-js/store";

const createCountStore = () => {
  const [state, setState] = createStore({ count: 0 });

  return {
    state: state,
    increment: () => setState(({ count }) => ({ count: count + 1})),
    // ...can also be written as
    // increment: () => setState("count", (count) => count + 1),
    decrement: () => setState(({ count }) => ({ count: count - 1}))
    // ...can also be written as
    // decrement: () => setState("count", (count) => count - 1)
  }
}

const store = createCountStore();

const CountDisplay = () => {
  return <p>Count: {store.state.count}</p>;
};

const CountIncrement = () => {
  return (
    <button onClick={() => store.increment()}>
      Increment
    </button>
  );
};

const CountDecrement = () => {
  return (
    <button onClick={() => store.decrement()}>
      Decrement
    </button>
  );
};


export default function App() {
  return (
    <>
      <CountDisplay />
      <CountIncrement />
      <CountDecrement />
    </>
  );
}
```

References

https://hackmd.io/@fendaya/rkMxmLRR_#React-Vue-3-Svelte-and-SolidJS-Cheatsheets