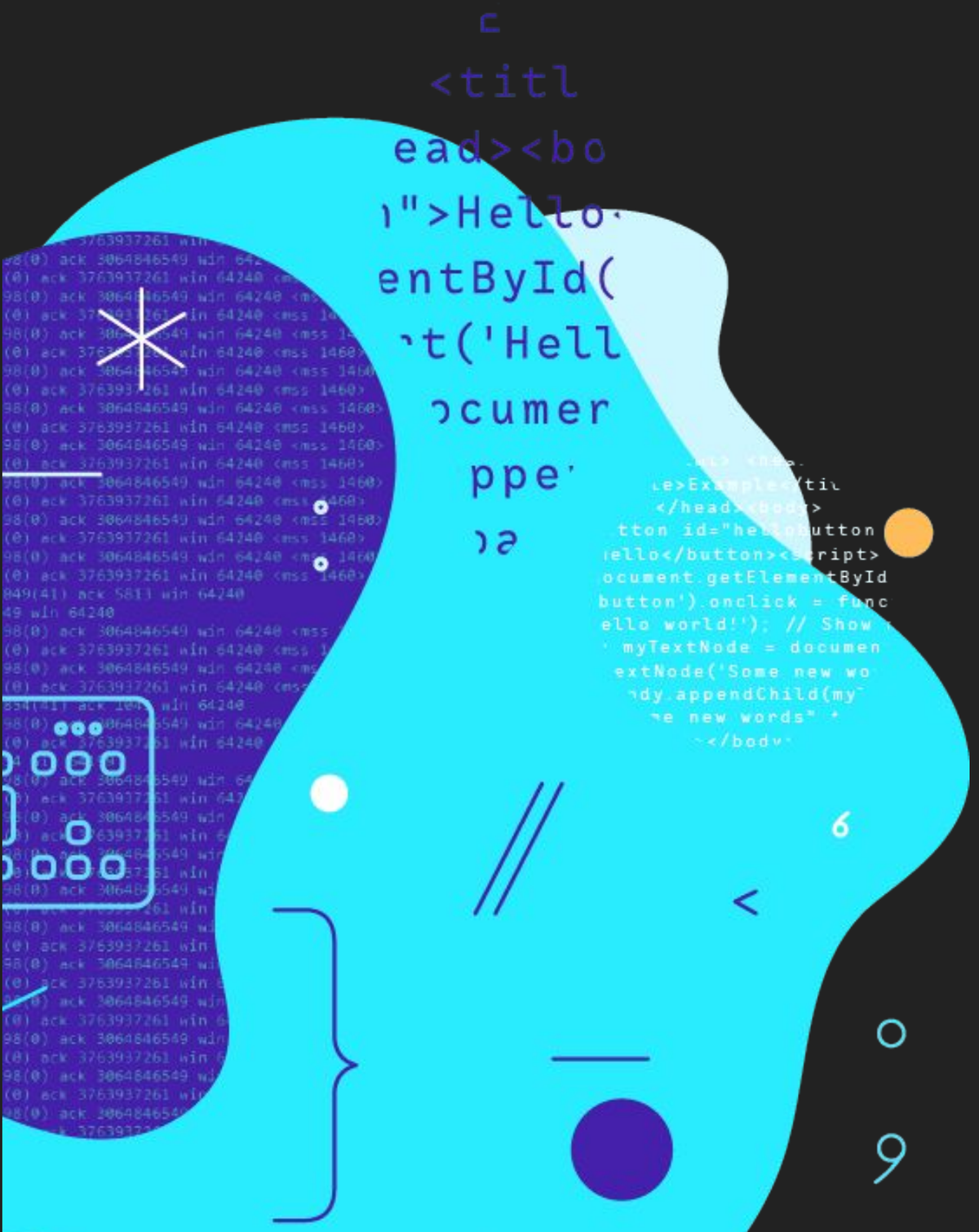
PHASE 3

WEEK 1

DAY 2



ELBRUS/BOOTCAMP

ПЛАН

1. Пользовательские хуки
2. memo
3. useContext
4. useMemo
5. useCallback
6. useReducer
7. CSS modules & SCSS

Пользовательские хуки

Пользовательские хуки

```jsx
// PokemonList.js
import React from 'react';
import Pokemon from './Pokemon';
import usePokemons from './hooks/use-pokemons';

function PokemonList({ pokemons }) {
  const [pokemonsData, loading, error] = usePokemons(pokemons);
  if (error) {
    return 'Ошибка загрузки покемонов.';
  }
  if (loading) {
    return 'Загрузка покемонов...';
  }
  return pokemonsData.map(({ name, weight, img }) => (
    <Pokemon key={name} name={name} weight={weight} img={img} />
  ));
}

export default PokemonList;

// hooks/use-pokemons.js
import { useState, useEffect } from 'react';
import fetchAll from '../lib/fetch-all';

function usePokemons(pokemons) {
  const [pokemonsData, setPokemonsData] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(false);

  useEffect(() => {
    if (!pokemons.length) {
      return false;
    }
    setLoading(true);
    setError(false);
    const abortController = new AbortController();
    const { signal } = abortController;
    (async () => {
      let jsonResults;
      try {
        jsonResults = await fetchAll(
          pokemons.map((name) => `https://pokeapi.co/api/v2/pokemon/${name}/`),
          { signal },
        );
      } catch (err) {
        setPokemonsData([]);
        setLoading(false);
        return setError(err);
      }
      setPokemonsData(jsonResults.map((data) => ({
        name: data.name,
        weight: data.weight,
        img: data.sprites.front_default,
      })));
      return setLoading(false);
    })();
    return () => abortController.abort();
  }, [pokemons]);

  return [pokemonsData, loading, error];
}

export default usePokemons;
```

memo

memo

Запрещает повторный рендеринг, если пропсы остались прежние.

Значительно повышает производительность при динамическом изменении списка компонентов.

```
// Pokemon.js
import React, { useState, memo } from 'react';

function Pokemon({ name, weight, img }) {
  const [
    currentWeight,
    setCurrentWeight,
  ] = useState(weight);
  function addWeight() {
    setCurrentWeight((x) => x + 50);
  }
  function removeWeight() {
    setCurrentWeight((x) => x - 50);
  }
  return (
    <>
      ...
    </>
  );
}

export default memo(Pokemon);
```

useContext

useContext

Как прокинуть данные в компоненты с глубокой вложенностью?

Как изменить данные в родительском компоненте?

Как повлиять на соседний компонент?

```
// contexts/pokemons-context.js
import { createContext } from 'react';

export default createContext({
  additionalWeight: 0,
  addAdditionalWeight: () => {},
  removeAdditionalWeight: () => {},
});
```

useContext (Provider)

```
// App.js
import React, { useState } from 'react';
import PokemonsContext from
'./contexts/pokemons-context';
import PokemonList from './PokemonList';
import PokemonsAdditionalWeight from
'./PokemonsAdditionalWeight';

function App() {
  const [
    additionalWeight,
    setAdditionalWeight,
  ] = useState(0);

  const addAdditionalWeight = (weight) => {
    setAdditionalWeight((x) => x + weight);
  };

  const removeAdditionalWeight = (weight) => {
    setAdditionalWeight((x) => x - weight);
  };

  return (
    <div className="App">
      <PokemonsContext.Provider
        value={{
          additionalWeight,
          addAdditionalWeight,
          removeAdditionalWeight,
        }}
      >
        <PokemonsAdditionalWeight />
        <PokemonList pokemons={['slowpoke',
'pikachu', 'psyduck']} />
      </PokemonsContext.Provider>
    </div>
  );
}

export default App;
```

useContext (Consumers)

```jsx
// PokemonsAdditionalWeight.js
import React, { useContext } from 'react';
import PokemonsContext from './contexts/pokemons-context';

function PokemonsAdditionalWeight() {
  const { additionalWeight } = useContext(PokemonsContext);
  return (
    <h2>
      Дополнительный вес покемонов:
      {additionalWeight}
    </h2>
  );
}

export default PokemonsAdditionalWeight;
```

```jsx
// Pokemon.js
import React, { useState, memo, useContext } from 'react';
import PokemonsContext from './contexts/pokemons-context';

const weightStep = 50;

function Pokemon({ name, weight, img }) {
  const { addAdditionalWeight, removeAdditionalWeight } =
useContext(PokemonsContext);
  const [currentWeight, setCurrentWeight] =
useState(weight);
  function addWeight() {
    setCurrentWeight((x) => x + weightStep);
    addAdditionalWeight(weightStep);
  }
  function removeWeight() {
    setCurrentWeight((x) => x - weightStep);
    removeAdditionalWeight(weightStep);
  }
  return (
    <>...</>
  );
}

export default memo(Pokemon);
```

useContext (нюанс)

Когда значение контекста меняется, все потребители рендерятся заново, независимо от memo.

useMemo

useMemo (вычисляем один раз)

```
// App.js
import React, { useState, useMemo } from 'react';
import PokemonsContext from './contexts/pokemons-context';
import PokemonList from './PokemonList';
import PokemonsAdditionalWeight from
'./PokemonsAdditionalWeight';

function App() {
  const [
    additionalWeight,
    setAdditionalWeight,
  ] = useState(0);

  const addAdditionalWeight = (weight) => {
    setAdditionalWeight((x) => x + weight);
  };

  const removeAdditionalWeight = (weight) => {
    setAdditionalWeight((x) => x - weight);
  };

  const pokemons = useMemo(
    () => ['slowpoke', 'pikachu', 'psyduck'],
    [],
  );

  return (
    <div className="App">
      <PokemonsContext.Provider
        value={{
          additionalWeight,
          addAdditionalWeight,
          removeAdditionalWeight,
        }}
      >
        <PokemonsAdditionalWeight />
        <PokemonList pokemons={pokemons} />
      </PokemonsContext.Provider>
    </div>
  );
}

export default App;
```

useMemo (борьба с useContext)

```
// Pokemon.js
import React, {
  useState, memo, useContext, useMemo,
} from 'react';
import PokemonsContext from './contexts/pokemons-context';

const weightStep = 50;

function Pokemon({ name, weight, img }) {
  const { addAdditionalWeight, removeAdditionalWeight } =
useContext(PokemonsContext);
  const [currentWeight, setCurrentWeight] = useState(weight);
  function addWeight() {
    setCurrentWeight((x) => x + weightStep);
    addAdditionalWeight(weightStep);
  }
  function removeWeight() {
    setCurrentWeight((x) => x - weightStep);
    removeAdditionalWeight(weightStep);
  }
  return useMemo(
    () => (
      <>
        <h2>{name}</h2>
        <strong>
          Вес:
          {' '}
          {currentWeight}
          {' '}
          гектограмм
        </strong>
        <button onClick={addWeight} type="button">
          Потолстеть
        </button>
        <button onClick={removeWeight} type="button">
          Похудеть
        </button>
        <img
          width={96 * (currentWeight / weight)}
          src={img}
          alt={name}
          style={{ display: 'block' }}
        />
      </>
    ),
    [name, weight, img, currentWeight, addWeight, removeWeight],
  );
}

export default memo(Pokemon);
```

useCallback

useCallback (борьба с useContext)

```jsx
// App.js
import React, { useState, useMemo, useCallback } from 'react';
import PokemonsContext from './contexts/pokemons-context';
import PokemonList from './PokemonList';
import PokemonsAdditionalWeight from './PokemonsAdditionalWeight';

function App() {
  const [
    additionalWeight,
    setAdditionalWeight,
  ] = useState(0);

  const addAdditionalWeight = useCallback((weight) => {
    setAdditionalWeight((x) => x + weight);
  }, []);

  const removeAdditionalWeight = useCallback((weight) => {
    setAdditionalWeight((x) => x - weight);
  }, []);

  const pokemons = useMemo(
    () => ['slowpoke', 'pikachu', 'psyduck'],
    [],
  );

  return (
    ...
  );
}

export default App;
```

```jsx
// Pokemon.js
import React, {
  useState, memo, useContext, useMemo, useCallback,
} from 'react';
import PokemonsContext from './contexts/pokemons-context';

const weightStep = 50;

function Pokemon({ name, weight, img }) {
  const { addAdditionalWeight, removeAdditionalWeight } =
useContext(PokemonsContext);
  const [currentWeight, setCurrentWeight] = useState(weight);
  const addWeight = useCallback(() => {
    setCurrentWeight((x) => x + weightStep);
    addAdditionalWeight(weightStep);
  }, [addAdditionalWeight]);
  const removeWeight = useCallback(() => {
    setCurrentWeight((x) => x - weightStep);
    removeAdditionalWeight(weightStep);
  }, [removeAdditionalWeight]);
  return useMemo(
    () => (
      <>
        ...
      </>
    ),
    [name, weight, img, currentWeight, addWeight, removeWeight],
  );
}

export default memo(Pokemon);
```

useCallback

useCallback + useMemo позволяют организовать точечный рендеринг только тех компонентов, которые реально поменялись, даже если используется useContext.

useReducer

Reducer && useReducer

```
// contexts/pokemons-context.js
import { createContext } from 'react';

export function reducer(state, action) {
  switch (action.type) {
    case 'addAdditionalWeight':
      return {
        ...state,
        additionalWeight: state.additionalWeight + action.weight,
      };
    case 'removeAdditionalWeight':
      return {
        ...state,
        additionalWeight: state.additionalWeight - action.weight,
      };
    default:
      return state;
  }
}

export default createContext({
  additionalWeight: 0,
  dispatch: () => {},
});
```

```
// App.js
import React, { useReducer } from 'react';
import PokemonsContext, { reducer } from './contexts/pokemons-context';
import PokemonList from './PokemonList';
import PokemonsAdditionalWeight from './PokemonsAdditionalWeight';

const pokemons = ['slowpoke', 'pikachu', 'psyduck'];

function App() {
  const [state, dispatch] = useReducer(
    reducer,
    { additionalWeight: 0 },
  );

  return (
    <div className="App">
      <PokemonsContext.Provider
        value={{
          ...state,
          dispatch,
        }}
      >
        <PokemonsAdditionalWeight />
        <PokemonList pokemons={pokemons} />
      </PokemonsContext.Provider>
    </div>
  );
}

export default App;
```

На стороне потребителя

```jsx
// Pokemon.js
import React, {
  useState, memo, useContext, useMemo, useCallback,
} from 'react';
import PokemonsContext from './contexts/pokemons-context';

const weightStep = 50;

function Pokemon({ name, weight, img }) {
  const { dispatch } = useContext(PokemonsContext);
  const [currentWeight, setCurrentWeight] = useState(weight);
  const addWeight = useCallback(() => {
    setCurrentWeight((x) => x + weightStep);
    dispatch({
      type: 'addAdditionalWeight',
      weight: weightStep,
    });
  }, [dispatch]);
  const removeWeight = useCallback(() => {
    setCurrentWeight((x) => x - weightStep);
    dispatch({
      type: 'removeAdditionalWeight',
      weight: weightStep,
    });
  }, [dispatch]);
  return useMemo(
    ...
  );
}

export default memo(Pokemon);
```

```jsx
// PokemonsAdditionalWeight.js
import React, { useContext } from 'react';
import PokemonsContext from './contexts/pokemons-context';

function PokemonsAdditionalWeight() {
  const { additionalWeight } = useContext(PokemonsContext);
  return (
    <h2>
      Дополнительный вес покемонов:
      {additionalWeight}
    </h2>
  );
}

export default PokemonsAdditionalWeight;
```

useReducer

Вместо множества функций — dispatch.

Логика изменения данных находится в редьюсере.

Не нужен useCallback на стороне провайдера, т.к. dispatch не создаётся заново.

CSS modules
SCSS

CSS modules

```css
/* Pokemon.module.css */
.pokemonContainer {
  width: 40%;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  align-items: stretch;
}

.pokemonContainer img {
  order: -1
}

.pokemonContainer h2 {
  order: -2
}
```

```jsx
// Pokemon.js
import React, {
  useState, memo, useContext, useMemo, useCallback,
} from 'react';
import PokemonsContext from
'./contexts/pokemons-context';
import styles from './Pokemon.module.css';

function Pokemon({ name, weight, img }) {
  ...
  return useMemo(
    () => (
      <div className={styles.pokemonContainer}>
        ...
      </div>
    ),
    [name, weight, img, currentWeight, addWeight,
removeWeight],
  );
}

export default memo(Pokemon);
```

SCSS

```scss
/* Pokemon.module.scss */
.pokemonContainer {
  width: 40%;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  align-items: stretch;
  img {
    order: -1
  }
  h2 {
    order: -2
  }
}
```

```jsx
// Pokemon.js
import React, {
  useState, memo, useContext, useMemo, useCallback,
} from 'react';
import PokemonsContext from
'./contexts/pokemons-context';
import styles from './Pokemon.module.scss';

function Pokemon({ name, weight, img }) {
  ...
  return useMemo(
    () => (
      <div className={styles.pokemonContainer}>
        ...
      </div>
    ),
    [name, weight, img, currentWeight, addWeight,
removeWeight],
  );
}

export default memo(Pokemon);
```