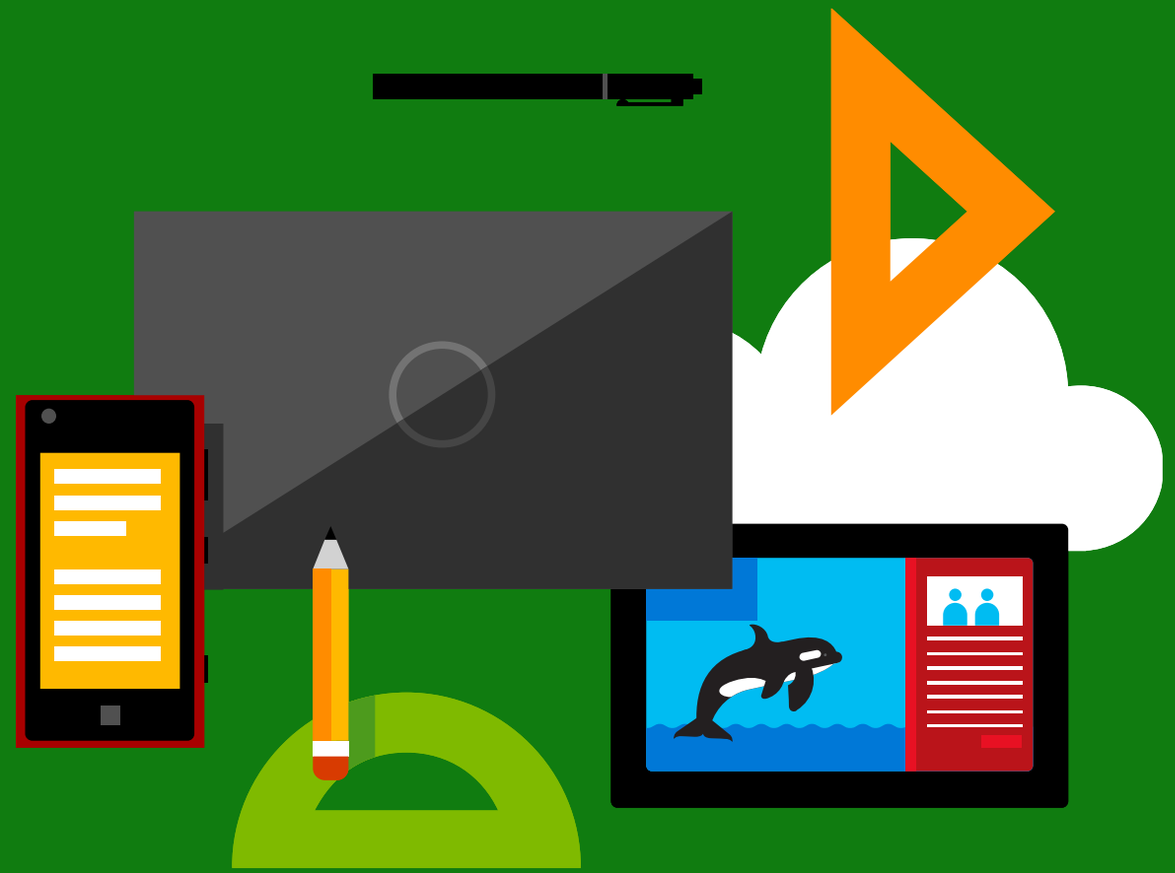
Desenvolvimento Web

HTML, CSS e Vue.js

Instrutor: Júlio Pereira Machado (julio.machado@pucrs.br)

Vue.js (continuação)



Composable

- “Composable” é uma função que encapsula e reutiliza lógica baseada em estado
- Permite implementar e compartilhar funcionalidade entre vários componentes
- Regra de nomenclatura:
  - Funções começam com a palavra “use”
- CUIDADO:
  - Composable são chamados de forma síncrona dentro de <script setup>
- Coleção de composables: https://vueuse.org/

Composable

Arquivo “mouse.js”

```
import { ref, onMounted, onUnmounted } from 'vue'

export function useMouse() {
  const x = ref(0)
  const y = ref(0)

  function update(event) {
    x.value = event.pageX
    y.value = event.pageY
  }

  onMounted(() => window.addEventListener('mousemove', update))
  onUnmounted(() => window.removeEventListener('mousemove', update))

  return { x, y }
}
```

Composable

Arquivo “App.vue”

```
<script setup>
import { useMouse } from './mouse.js'

const { x, y } = useMouse()
</script>

<template>
  Mouse position is at: {{ x }}, {{ y }}
</template>
```

Composable

- Composable devem ser flexíveis e suportar argumentos com *ref* e funções *getter*
- Utilizar a função *toValue()* para processar os diferentes tipos de argumentos e retornar o valor associado

```
import { toValue } from 'vue';

function useFeature(talvezRefOrGetter) {
  // se talvezRefOrGetter é um ref ou um getter,
  // seu valor normalizado será retornado.
  // Caso contrário, retorna diretamente.
  const value = toValue(talvezRefOrGetter);
}
```

Roteamento

- Em uma Single-Page Application (SPA), código JavaScript no lado cliente é capaz de interceptar as ações de navegação e decidir sobre a *view* sendo renderizada
- O roteador "oficial" é o Vue Router

https://router.vuejs.org/

Vue Router

- Funcionalidades:
  - Mapeamento de rotas aninhadas
  - Roteamento dinâmico
  - Configuração de rotas com base em componentes
  - Suporte a *route params*, *query* e curingas
  - Aplicação de transições animadas durante navegação
  - Links com CSS dinâmicos
  - Suporte aos modos *history/hash* do HTML5
  - Customização do comportamento de scrolling
  - Codificação de URLs

Vue Router

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      component: HomeView
    },
    {
      path: '/about',
      name: 'about',
      component: () => import('../views/AboutView.vue')
    }
  ]
})

export default router
```

Vue Router

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(router)

app.mount('#app')
```

Vue Router

- Parâmetros em rotas são denotados por *:chave*

Rota	Significado
{ path: '/o/:orderId' }	matches /o/3549
{ path: '/p/:productName' }	matches /p/books
{ path: '/:orderId(\\d+)' }	/:orderId -> matches only numbers
{ path: '/:chapters+' }	/:chapters -> matches /one, /one/two, /one/two/three, etc
{ path: '/:chapters*' }	/:chapters -> matches /, /one, /one/two, /one/two/three, etc
{ path: '/users/:userId?' }	will match /users and /users/posva

Vue Router

- Componente <RouterLink> é responsável por renderizar uma referência <a> no HTML
  - Link de destino é especificado na propriedade *to*
- Componente <RouterView> é reponsável por renderizar o componente associado à URL navegada

Vue Router

```
<template>
  <header>

    <div class="wrapper">
      <HelloWorld msg="You did it!" />

      <nav>
        <RouterLink to="/">Home</RouterLink>
        <RouterLink to="/about">About</RouterLink>
      </nav>

    </div>
  </header>

  <RouterView />

</template>
```

Vue Router

- Acesso ao roteador/rota estará disponível para o componente através das funções *useRouter()* e *useRoute()*
- Dentro de <template> pode ser utilizado diretamente *$router* e *$route*

```
import { useRoute } from 'vue-router'
import { ref, watch } from 'vue'

export default {
  setup() {
    const route = useRoute()
    const userData = ref()
    // fetch the user information when params change
    watch(
      () => route.params.id,
      async newId => {
        userData.value = await fetchUser(newId)
      }
    )
  },
}
```

Vue Router

- Navegação pode realizada de forma programática
- Para navegar para uma nova URL, utiliza-se método assíncrono *router.push()*

```
// literal string path
router.push('/users/eduardo')

// object with path
router.push({ path: '/users/eduardo' })

// named route with params to let the router build the url
router.push({ name: 'user', params: { username: 'eduardo' } })

// with query, resulting in /register?plan=private
router.push({ path: '/register', query: { plan: 'private' } })

// with hash, resulting in /about#team
router.push({ path: '/about', hash: '#team' })
```

Vue Router

- Para navegar para uma nova URL, sem manter histórico de navegação, utiliza-se método assíncrono *router.replace()*

```
router.push({ path: '/home', replace: true })
// é equivalente a
router.replace({ path: '/home' })
```