

# PROGRAMAÇÃO ORIENTADA A OBJETOS

---

Alessandro Valério e Edson Ifarraguirre Moreno

**“ Não adianta você ser um bom programador,  
mas você não saber o que o **mercado** precisa.”**

*Juliana Remor*

# Conheça o livro da disciplina

## CONHEÇA SEUS PROFESSORES 3

*Conheça os professores da disciplina.*

## EMENTA DA DISCIPLINA 4

*Veja a descrição da ementa da disciplina.*

## BIBLIOGRAFIA DA DISCIPLINA 5

*Veja as referências principais de leitura da disciplina.*

## O QUE COMPÕE O MAPA DA AULA? 6

*Confira como funciona o mapa da aula.*

## MAPA DA AULA 7

*Veja as principais ideias e ensinamentos trabalhados ao longo da aula.*

## RESUMO DA DISCIPLINA 28

*Relembre os principais conceitos da disciplina.*

## AVALIAÇÃO 29

*Veja as informações sobre o teste da disciplina.*

# Conheça seus professores



## ALESSANDRO VALÉRIO DIAS

Professor Convidado

---

Mestre em Administração e Negócios pela PUCRS, possui também os títulos de especialista em duas áreas: Gerenciamento de Projetos de Tecnologia da Informação e Informática na Educação. É bacharel em Ciência da Computação e Psicologia pela Pontifícia Universidade Católica do Rio Grande do Sul. Atualmente, é analista de sistemas da Pontifícia Universidade Católica do Rio Grande do Sul e professor dos cursos de Análise e Desenvolvimento de Sistemas e de Ciência da Computação do Centro Universitário UniRitter.

## EDSON IFARRAGUIRRE MORENO

Professor PUCRS

---

Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Atualmente, é professor adjunto pela mesma PUCRS, estando vinculado a Escola Politécnica, sendo responsável por disciplinas da área de hardware para os cursos de Ciência da Computação, e Engenharia da Computação. Adicionalmente, trabalha com a orientação de alunos no desenvolvimento de projetos do curso de Engenharia de Software. Desde 2016, coordena o laboratório iSeed Labs, uma parceria entre a academia e a iniciativa privada que tem por objetivo fomentar a inovação e o empreendedorismo. Seus principais temas de pesquisa incluem: sistemas multiprocessados em chip (Multiprocessor System on Chip, MPSoC), projeto em nível de sistema e redes em chip (Network on Chip, NoC).



# *Ementa da Disciplina*

Estudo sobre conceitos de Classes (atributos, métodos, propriedades, visibilidade, instancia ou classe). Estudo de conceitos de Herança, Polimorfismo, Interfaces, Genéricos e Arrow functions. Estudo sobre funções de filtragem, mapeamento e redução. Estudo sobre construtores de tipos.

# Bibliografia da disciplina

As publicações destacadas têm acesso gratuito.

## Bibliografia básica

GRONER, Loiane. **Learning JavaScript Data Structures and Algorithms**. Birmingham: Packt, 2018.

HAVERBAKE, Marjin. **Eloquent JavaScript**. São Francisco: No Starch Press, 2019.

WEISFELD, Matt. **The Object Oriented Thought Process**. Boston: Addison-Wesley, 2019.

## Bibliografia complementar

MELONI, Julie C. Sams. **Teach Yourself HTML, CSS and JavaScript All in One**. Carmel: Sams Publishing, 2019.

FLANANGAN, David. **JavaScript**: the definitive guide. Sebastopol: O'Reilly, 2020.

FREEMAN, Steve; PRYCE, Nat. **Growing Object Oriented Software, Guided by Tests**. Boston: Addison-Wesley, 2010.

ROZENTALS, Nathan. **Mastering TypeScript**. Birmingham: Packt, 2021.

ZAKAS, Nicholas C. **Professional JavaScript for web developers**. Indianapolis: John Wiley & Sons, 2019.

# O que compõe o Mapa da Aula?

## MAPA DA AULA

São os capítulos da aula, demarcam momentos importantes da disciplina, servindo como o norte para o seu aprendizado.



## EXERCÍCIOS DE FIXAÇÃO

Questões objetivas que buscam reforçar pontos centrais da disciplina, aproximando você do conteúdo de forma prática e exercitando a reflexão sobre os temas discutidos. Na versão online, você pode clicar nas alternativas.



## PALAVRAS-CHAVE

Conceituação de termos técnicos, expressões, siglas e palavras específicas do campo da disciplina citados durante a videoaula.



## VÍDEOS

Assista novamente aos conteúdos expostos pelos professores em vídeo. Aqui você também poderá encontrar vídeos mencionados em sala de aula.



## PERSONALIDADES

Apresentação de figuras públicas e profissionais de referência mencionados pelo(a) professor(a).



## LEITURAS INDICADAS

A jornada de aprendizagem não termina ao fim de uma disciplina. Ela segue até onde a sua curiosidade alcança. Aqui você encontra uma lista de indicações de leitura. São artigos e livros sobre temas abordados em aula.



## FUNDAMENTOS

Conteúdos essenciais sem os quais você pode ter dificuldade em compreender a matéria. Especialmente importante para alunos de outras áreas, ou que precisam relembrar assuntos e conceitos. Se você estiver por dentro dos conceitos básicos dessa disciplina, pode tranquilamente pular os fundamentos.

## CURIOSIDADES

Fatos e informações que dizem respeito a conteúdos da disciplina.



## DESTAQUES

Frases dos professores que resumem sua visão sobre um assunto ou situação.



## ENTRETENIMENTO

Inserções de conteúdos para tornar a sua experiência mais agradável e significar o conhecimento da aula.



## CASE

Neste item, você relembra o case analisado em aula pelo professor.



## MOMENTO DINÂMICA

Aqui você encontra a descrição detalhada da dinâmica realizada pelo professor.



# Mapa da Aula

Os tempos marcam os principais momentos das videoaulas.

## AULA 1 • PARTE 1

“ Os objetos são uma estrutura um pouco mais complexa que as funções. ”

### PALAVRA-CHAVE

**Framework:** Em desenvolvimento de software, o framework é uma abstração que une códigos comuns entre vários projetos de software, assim provendo uma funcionalidade genérica. Em outras palavras, um framework de software compreende de um conjunto de classes implementadas em uma linguagem de programação.

“ Quem programa tem que saber ler código, é uma habilidade que a gente tem que desenvolver ao longo do tempo, como gostar de ver código bonito. ”

### Conceitos I

a) **Programação estruturada:** surgida nos anos 1940, com as primeiras linguagens de caráter algorítmico, as quais visavam resolver problemas específicos, é um paradigma com foco em sequência (de itens, comandos, dados...), em decisão

04:57



#### Introdução

A Programação Orientada a Objetos (POO) é um paradigma de programação com foco em objetos, ao invés de funções. Ainda, a POO não pode ser considerada uma linguagem de programação, tampouco uma ferramenta ou um framework. É, na verdade, um estilo de programação.

São diversas as linguagens de programação que implementam a orientação a objetos, como C++, C#, Java, JavaScript, Python e Ruby. Vale destacar que nenhuma dessas linguagens é puramente orientada a objetos.

10:01



“ Muitas linguagens, atualmente, que estão em alta no mercado ou ao menos que estão bem estabelecidas, implementam ou dão suporte à orientação a objetos. ”

13:57



“ Qualquer bom profissional de desenvolvimento de software deve conhecer paradigmas. ”

14:42



(testes lógicos) e em iteração (repetição);

b) **Programação procedural:** paradigma com foco no uso de procedimentos e funções para facilitar o reuso, estão presentes não só os elementos da programação estruturada, mas um conjunto de funções.

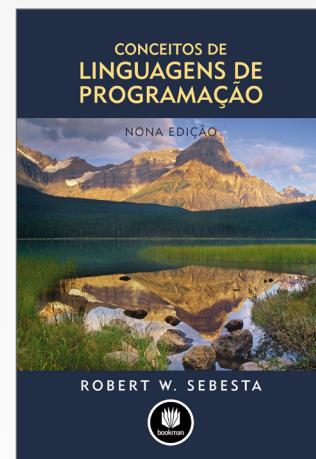
O maior dilema da programação estruturada e da programação procedural é que o elevado número de funções, em meio às quais não há qualquer senso de pertencimento, e a falta de isolamento de alguns dados gerou uma série de problemas. Havia interdependência de funções, muito “cópia e cola” de funções, as mudanças em uma função resultavam em mudanças em outras funções...

16:58



## LEITURA INDICADA

### Conceitos de linguagens de programação



Publicado em 2011 por Robert W. Sebesta, o livro apresenta as principais construções das linguagens de programação contemporâneas e oferece as ferramentas necessárias para uma avaliação crítica das linguagens existentes e futuras.



## LEITURA INDICADA

### Código limpo

19:49



24:31



## PALAVRA-CHAVE

**Go to:** O comando goto, do inglês “go to” (“ir para”, em português) é uma estrutura de controle para salto de instruções muito usada em programação estruturada, baseada em sequências. Sua sintaxe é, em geral, “goto destino”, de forma que as instruções passam a ser executadas no endereço apontado por destino.

34:26



“A gente vai passar 75% do tempo dando manutenção em códigos, não fazendo códigos novos.”

38:33



“Esta é outra habilidade que desenvolvedores bons têm que ter: paciência para ler documentação.”

Publicado em 2008 por Robert Cecil Martin, o livro apresenta um paradigma revolucionário, oferecendo práticas melhores e mais ágeis para limpar códigos “dinamicamente”. A obra é essencial para qualquer desenvolvedor, engenheiro de software, gerente de projeto, líder de equipes ou analistas de sistemas com interesse em construir códigos melhores.





## EXERCÍCIO DE FIXAÇÃO

Surgido nos anos 1940, com as primeiras linguagens de caráter algorítmico, as quais visavam resolver problemas específicos, é um paradigma com foco em sequência e iteração. Assinale a alternativa que corresponda corretamente à afirmação.

## AULA 1 • PARTE 2

00:24



### Conceitos II

c) **Programação Orientada a Objetos:** trata-se de um paradigma como foco no uso de objetos, em que cada um contém suas próprias variáveis, funções (métodos) e conjuntos de dados (atributos). Esses objetos, uma construção a partir da qual podemos criar elementos semelhantes, possuem uma série de componentes: um nome único, atributos/variáveis e comportamentos/métodos.

03:03



05:00



06:56



10:26



## PALAVRA-CHAVE

**UML:** Aprovada como padrão pelo OMG (Object Management Group) em 2000, a Unified Modeling Language é uma linguagem-padrão para a elaboração da estrutura de projetos de software. A UML pode ser empregada na visualização, especificação, construção e documentação de artefatos que façam uso de sistemas complexos de software.

“ Tudo o que é dado de um objeto, a gente chama de atributo ou propriedade. ”

“ Um objeto pode ter nenhuma variável, pode ter zero ou inúmeras variáveis. E o objeto também pode ter nenhuma lógica, função ou método, e pode ter inúmeras. ”

“ Os atributos, as informações que estão dentro de um atributo representam o estado daquele objeto: quem ele é, o que ele representa... E os métodos representam o que o objeto pode fazer. ”

## Objeto

Um objeto pode ser definido como uma coleção de dados e/ou funcionalidades com alguma relação entre si. Vale sublinhar que dados, variáveis, atributos e propriedades podem ser considerados sinônimos no contexto da Programação Orientada a Objetos.

Em JavaScript, linguagem utilizada pelo professor ao longo das suas duas aulas, os atributos e os métodos são normalmente membros de um objeto, de forma que cada um tenha um nome e um valor.

11:28



17:21

“ Um objeto, em JavaScript, é composto de membros que têm o nome e o valor do membro. ”

24:01



## Atributo ou propriedade

Trata-se de um ou mais dados que estão presentes em um objeto. A título de exemplo, através do console, o professor cria dois objetos representando duas pessoas diferentes, as quais possuem as suas próprias variáveis e seus próprios métodos. Os atributos e propriedades possuem um nome único e armazenam um valor ou uma referência.

## AULA 1 • PARTE 3

## Acesso e atribuição

O acesso às propriedades ou atributos de um objeto se dá por meio do comando `nomeObjeto.nomePropriedade`. O acesso aos vetores, por sua vez, se dá pelo comando `nomeObjeto["nomePropriedade"]`.

Para mudar o valor de uma propriedade, é preciso executar uma atribuição, que pode ser feita de duas formas: `nomeObjeto.nomePropriedade = algo`, ou `nomeObjeto["nomePropriedade"] = algo`.

00:24



07:09

“ Atributo, propriedade e variável são sinônimos neste caso. ”

07:25



## Métodos

O método representa uma ou mais funcionalidades presentes em um objeto. Assim como os atributos, os métodos possuem nome único. Além disso, representam também uma lógica pertinente ao objeto. A única forma de acessar um método é por meio do comando `nomeObjeto.metodo()`. Se houver parâmetros, usa-se `nomeObjeto.metodo(parametro)`. Assim como no caso das propriedades, podemos mudar um método através da atribuição.

16:04



“ O primeiro jeito de criar objetos é como fizemos até agora: a gente vai criar de forma literal - como é, como está sendo lido. ”

## Comparando

Em comparação ao modelo da programação estruturada, a POO nos permite agrregar dados em objetos, de formas que as variáveis e as funções façam parte desses objetos. Há mudança na forma como enxergamos a informação.

## Tipos de dados

Em JavaScript, é possível mudar o domínio de uma variável – isto é, seus valores – ao longo da execução de um programa. Por essa razão, inclusive, essa linguagem é considerada fracamente tipada.

Os dados de JavaScript podem ser segmentados em dois grandes tipos: **de valor (ou primitivos) e de referência (ou definidos pelo usuário)**. No primeiro caso, com os tipos Number, String, Boolean, Symbol, undefined e null, falamos de representações de valores imutáveis. No segundo caso, falamos de representações de valores mutáveis e complexos.

## PALAVRA-CHAVE

**PCI e PII:** Informações de identificação pessoal, ou “personally identifiable information” (PII), e dados do setor de cartões de pagamento, ou “payment card industry” (PCI), são categorias de informação usadas pelas organizações para identificar indivíduos e fornecer a eles determinados serviços. Tais categorias, nesse sentido, se enquadram no âmbito da governança da informação.



20:31



26:39



26:57



39:51



40:15



41:59



*Funções são objetos em JavaScript, dados complexos (que são os próprios objetos) são objetos em JavaScript, estruturas de dados são objetos, arrays são objetos...*



*A gente tá sempre abstraindo coisas do mundo real pra transformar em um programa.*



## Encapsulamento

O encapsulamento permite que atributos e métodos sejam agrupados de certa forma em uma interface bem definida a fim de manipular os dados de um objeto de forma eficiente. O objeto e os atributos, nesse sentido, precisam ser protegidos expondo para o mundo externo apenas aquilo que o desenvolvedor julgar necessário.

Há benefícios e perigos associados ao encapsulamento. No primeiro caso, como já foi exposto, o objeto pode ser protegido, de forma que informações sensíveis não sejam livremente alteradas e a integridade do programa, comprometida.

## EXERCÍCIO DE FIXAÇÃO



Trata-se de um paradigma como foco no uso de objetos, em que cada um contém suas próprias variáveis, funções (métodos) e conjuntos de dados (atributos). Assinale a alternativa que corresponda corretamente à afirmação.

## AULA 1 • PARTE 4

00:24



### Abstração

01:59



O exercício constante de tentar entender o problema a ser resolvido e como implementar a solução chama-se “abstração”. Diferentemente da programação estruturada e da programação procedural, que ainda assim possuem certo grau de abstração, a POO é amplamente baseada na abstração digital da vida real.

03:48



Objetos são abstrações daquilo que desejamos implementar, tendo em vista o que observamos no mundo ao nosso redor. Nesse sentido, buscamos o essencial e deixamos de lado o que não importa, focando naquilo que realmente precisamos representar em nossos objetos.



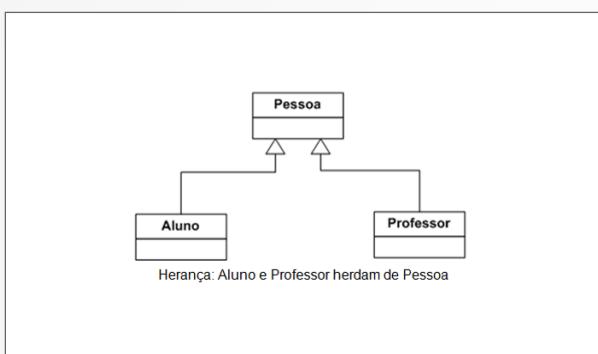
05:34

### Herança

Permite o compartilhamento de atributos e métodos entre os objetos, reutilizando código e agrupando o que há de comum a diferentes objetos. Além disso, a herança busca identificar e agrupar comportamentos generalizados ou especialização, e ajuda a eliminar

redundâncias.

No caso de uma plataforma, por exemplo, podemos ter um objeto de nome “usuario” ao qual estão ligados atributos como identificação, e-mail e senha, além de funções como fazer login, trocar e-mail e trocar a senha. Um outro objeto, de nome “administrador”, pode ser conectado a esse objeto inicial, mas associado a funções que são pertinentes a ele. A elas, porém, são somadas as funções pertencentes ao primeiro objeto. É a herança a responsável por estabelecer esses elementos em comum.



13:06



“ Tudo o que conseguimos de informações que podemos generalizar vão subindo nesta árvore. ”

19:23



## Polimorfismo

Por meio da herança, o polimorfismo nos permite alterar um comportamento herdado de um objeto-pai, o que torna possível uma forte separação de interesses, além de promover a limpeza do código, removendo a lógica excedente.

Já que, no polimorfismo, falamos de comportamento, a nossa preocupação deve ser orientada aos métodos. Nesse sentido, há duas formas de polimorfismo: a **sobrescrita de método**, quando um mesmo método (mesmo nome e mesma assinatura) é usado em dois objetos diferentes, e a **sobrecarga de método**, quando métodos com o mesmo nome e com diferentes assinaturas são relacionados através da herança a diferentes objetos.

“ O polimorfismo, junto com a herança, é a grande sacada da Programação Orientada a Objetos. ”

“ O JavaScript não suporta sobrecarga. ”

“ Pensem em economia. Gastem o tempo de vocês com mais estudo, com mais qualidade, testando melhor as coisas... Codar não é tudo, tem bastante coisa em volta do programar que também é importante. ”

20:05



31:06



## Vantagens

O encapsulamento nos oferece proteção de dados e redução da complexidade. A abstração, além de reduzir a complexidade, garante maior reuso dos objetos. A herança, por sua vez, é capaz de eliminar as redundâncias do código. E o polimorfismo, por fim, elimina lógicas desnecessárias no código.

32:44



35:08



## Criando objetos

Há diversas formas de criar objetos: a forma literal, quando o objeto é criado “na mão”, os elementos são definidos e associados a variáveis; as funções fabricantes, que são responsáveis por criar objetos; as funções construtoras, os protótipos e as classes.

45:52



39:26



## PALAVRA-CHAVE

**Syntax sugar:** Em computação, o chamado “açúcar sintático” é a sintaxe de uma linguagem de programação que visa tornar as suas construções mais fáceis de serem lidas. O nome vem da ideia de que, com essa otimização, o uso da linguagem torna-se “mais doce” para o uso humano, permitindo que suas funcionalidades sejam expressas mais clara e concisamente.

## AULA 2 • PARTE 1

### Construtores

Prosseguindo o raciocínio de criação de objetos, aborda-se agora as funções construtoras. Elas criam objetos de forma mais alinhada com a orientação objeto. Cita-se a nova palavra-chave “new”, ou seja, uma instância de objeto novo criado, construído pela própria função. Logo, a palavra-chave “new” cria um objeto baseado no construtor.

Nesse sentido, ressalta-se a necessidade de separar o que é um construtor e o que é um objeto. A palavra “new” aciona a própria execução do construtor em específico. Também destaca-se a necessidade de especificar os nomes durante a programação, a fim de colocar uma lógica significativa e clara. É fundamental dar nomes significativos tanto para variáveis quanto para parâmetros, métodos e fundamentais, na medida em que uma linguagem organizada auxilia olhares de outros que posam vir a trabalhar nas funções.

Dessa forma, destaca-se o “this” como uma referência ao próprio objeto, assim como o fato de, em JavaScript, a herança ocorrer através de protótipos.

02:37



01:50



04:21



Provavelmente, se vocês já fizeram qualquer código em JavaScript, vocês já usaram a palavra-chave ‘new’ para alguma coisa.



07:31



Esta função está criando o objeto em si como resultado da execução dessa função. Ela é uma função construtora, então, está construindo o objeto.



Eu estou fazendo uma atribuição. Então, por isso que não é vírgula no fim. Não estou criando um objeto com vários itens: eu estou já acessando os itens do objeto.



- 
- “ Todo objeto, na verdade, é enxergado como uma instância em orientação objeto. É bem importante que a gente se lembre disso. ”
- “ O medo de a gente dar manutenção passa também por uma fraca qualidade nos nomes das coisas que a gente dá. ”
- “ Função construtora é aquela que não te entrega um objeto. Ela é responsável por criar um objeto, sabe como fazer isso, e prescinde da palavra ‘new’. ”
- “ Toda vez que a gente precisa acessar algum membro do objeto dentro dele, a gente referencia ‘this’. Então, ‘this.este membro’, seja uma função, seja uma variável ou atributo. ”
- “ Mesmo que o construtor pessoa tenha mudado ao longo do tempo, os objetos guardam a memória dos seus construtores. Por que? Porque foi ele que construiu. ”
- 12:23
- 18:06
- 20:57
- 27:00
- 27:51
- 30:11
- 32:48
- 36:54
- “ Então, clareza e intenção na hora de programar. Coloquem nomes significativos, que deem todo o contexto para quem está programando. ”
- “ A palavra ‘new’ é a chave para acionar o construtor que está dentro dessa função. Lembrem que na função fábrica, o construtor era um pouco mais visível, ele aparecia. ”
- “ Como o usuário, no nosso exemplo aqui, não é filho de ninguém, ele não é órfão: ele é filho de object. Então, na verdade, todo mundo é filho de object por herança, direta ou indireta. ”



## EXERCÍCIO DE FIXAÇÃO

Assinale a alternativa incorreta.

### AULA 2 • PARTE 2

#### Protótipos

O protótipo é um mecanismo de herança entre objetos. A herança dá diversas vantagens, tal como a economia de códigos, o que é demonstrado a partir do “value of” contido no programa. Nesse sentido, o professor realiza vários testes com o construtor, a partir do domínio em relação às heranças, fundamentais para a qualidade do código. Conforme ressaltado, o Java Script é uma linguagem baseada em protótipos. Ademais, todo objeto pode ter um objeto de protótipo dentro dele, e todo objeto possui uma propriedade chamada “prototype”.

A herança também pode ser usada para alterar determinados objetos, inclusive objetos que possam vir a existir, o que facilita a criação manual de objetos, além de contribuir para um código limpo e organizado.



00:25

06:07



10:22



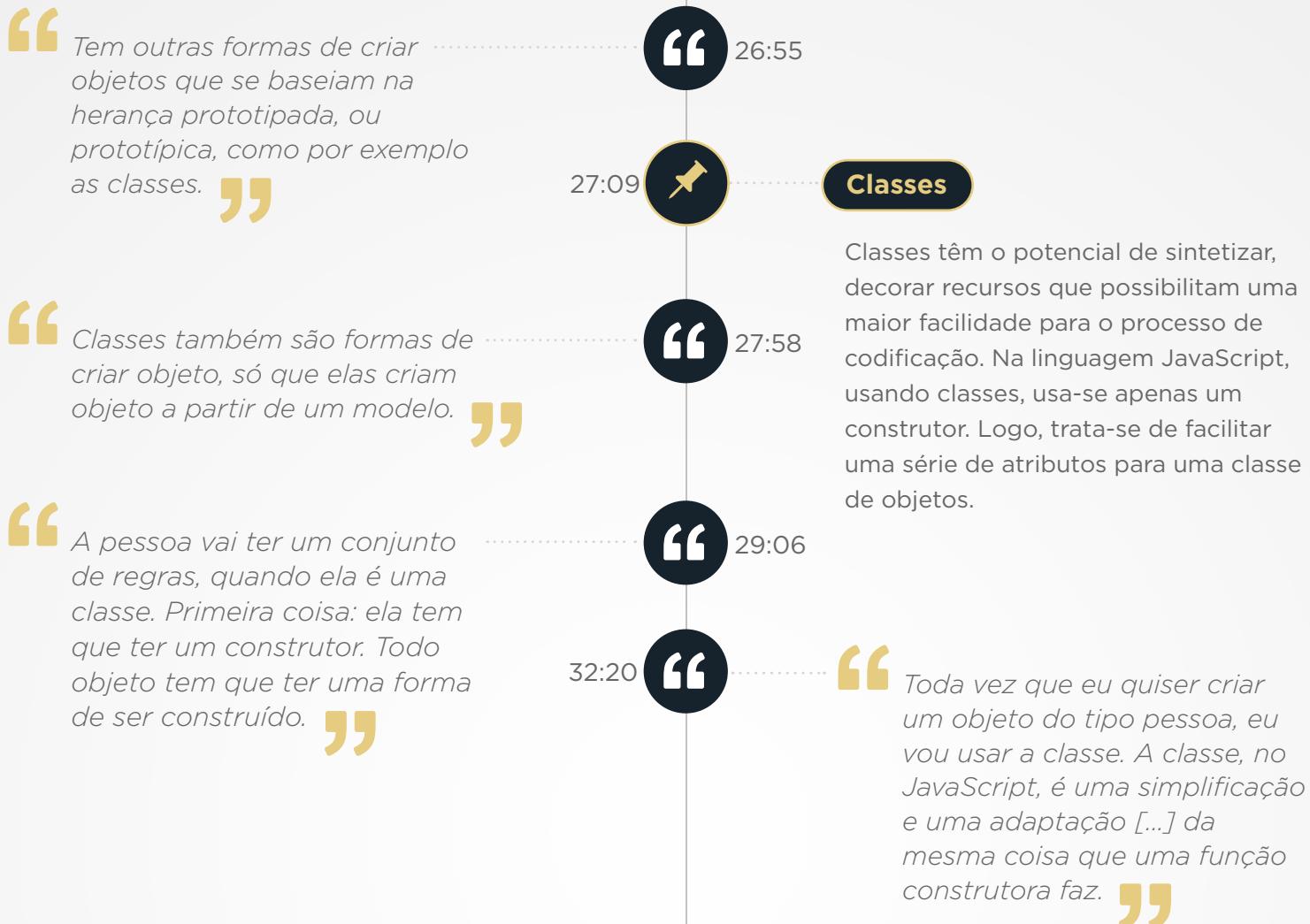
12:20



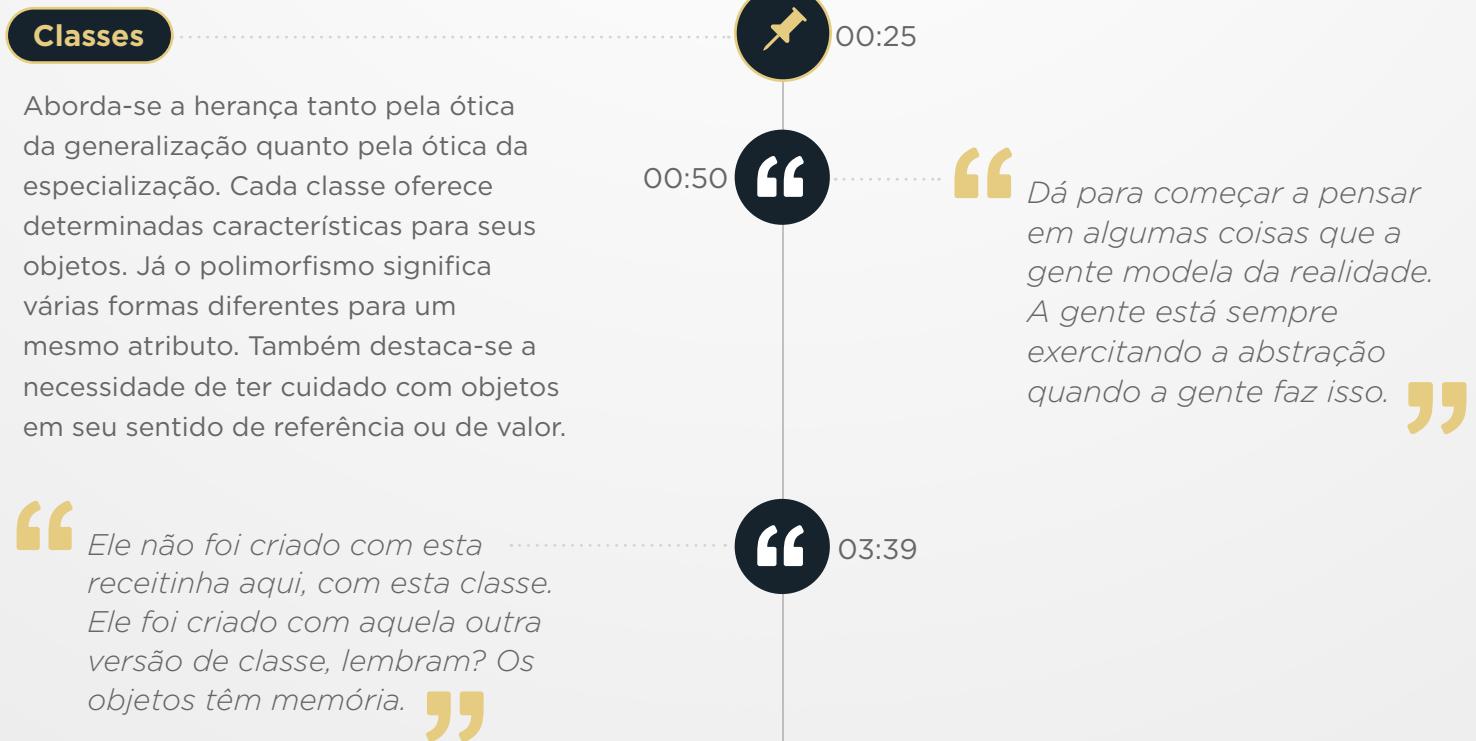
**“**Outra boa prática de programação: os valores estáticos, primeiro. **”**

**“**A qualidade do código também é isto: cuidar níveis de herança, se há necessidade mesmo de se criar objetos separados, ou se a visão que a gente precisa não é mais única. **”**

**“**O que para nós é uma visão de herança, para o JavaScript, a gente chama de cadeia de protótipos. **”**



## AULA 2 • PARTE 3



## Referência

Há de se ter cuidado na delimitação entre a cópia e a diferenciação de objetos, tendo em vista que as ideias em código podem se embaralhar na referênciação. Nesse sentido, ressalta-se a possibilidade de alterar atributos e métodos.

## Membros e suas propriedades

Ressalta-se a possibilidade de colocar métodos e funções estáticas, ambas testadas em aula no código. Os atributos não estáticos são aqueles que são parte do objeto. Em contraponto, os atributos estáticos são compartilhados por todas as instâncias de objeto.

## EXERCÍCIO DE FIXAÇÃO

Assinale a alternativa incorreta.

11:25



“

Quando a gente faz uma string igualar a outra, ou um number igualar o outro, a gente não está copiando aquele number, porque o number é um valor literal.

”

11:58



21:37



“

Toda vez que eu criar um professor, ele agora vai gerar com uma formação também, vai ter que completar depois. Eu estou mexendo no protótipo.

”

21:59



22:50



“

O elemento estático, o componente estático, ele é estático, não depende das instâncias de objeto. Ele é ligado diretamente ao protótipo ou à classe.

”

30:52



“

Se ele é estático, e não está protegido, eu consigo pegar ele e acessar ele de fora também, de qualquer outra classe.

”

## AULA 2 • PARTE 4

### Visibilidade

É fundamental saber as diferenças entre o público e o privado no interior dos códigos. É por essa razão que o JavaScript possui identificadores pré-fixados que dizem respeito à privacidade. Também destaca-se a necessidade de atenção às variáveis de instância e às variáveis locais.

Nesse sentido, ressalta-se a importância de perceber se estão sendo devolvidos ao usuário referências ou valores dos objetos. Pode ser perigoso deixar visível certos conteúdos na orientação objeto, ou seja: a privacidade dos dados é muito importante.



00:25



02:34



Uma variável local, se a gente pensar em funções, o escopo delas é local, então nada fora daquela função consegue acessar ela. É como se o ciclo de vida dela se encerrasse quando a função termina de executar.



A gente às vezes trabalha tanto e mexe tanto com as coisas que esquece os fundamentos. O bacana de poder revisitar isso e contar para alguém é que a gente também se atualiza com o passado.



A gente consegue proteger um objeto para não permitir que ele crie um novo atributo.



13:58



23:52

### Associação entre objetos

Trabalhar com orientação objeto é, também, trabalhar com relações entre objetos distintos. Logo, é preciso saber associar os objetos entre si, entre proteções, limitações e permissões.



Daqui a dez anos as linguagens talvez não estejam tão na moda como hoje. Então, a gente tem que poder sempre entender o conceito básico, usar naquela linguagem A, depois usar na B.



36:49



38:26



38:43



É importante ampliar o leque de vocês em relação a conhecimento técnico, não importa qual linguagem. Estudem outras linguagens. É tão bacana ser poliglota. E a gente é. Tem que ser.



## AULA 3 • PARTE 1

05:30



### Introdução

Reafirmando o conteúdo trabalhado

initialmente nas duas primeiras aulas da disciplina, o professor destaca que a orientação a objetos é um estilo de programação, não uma linguagem em si. As linguagens, na verdade, se apoiam no paradigma da Programação Orientada a Objetos. Suportada por diferentes tecnologias e linguagens, a POO é largamente empregada na indústria atualmente.

11:20



11:51



**“** [Em JavaScript], eu não preciso efetivamente declarar os tipos que eu estou usando para as diferentes formas de declarar uma variável. **”**

### PALAVRA-CHAVE

**Front-end:** Refere-se ao desenvolvimento da interface gráfica do utilizador de um website, realizado através do uso de HTML, de CSS ou de JavaScript. Nesse sentido, o desenvolvedor frontend é responsável pela experiência do usuário de uma determinada aplicação, definindo a disposição das páginas do site, o seu layout e a sua aparência.

### PALAVRA-CHAVE

**Back-end:** Refere-se à infraestrutura de funcionamento de uma determinada solução - isto é, dos recursos que possibilitam a operação de um sistema e que garantem a lógica da solução. Tudo aquilo que está por trás da interface gráfica do usuário está ligado ao back-end, que define recursos habilitadores da solução, soluções em nuvem e meios de autenticação.

**“** Eu posso modificar o valor associado a um objeto que está associado à minha constante, mas não posso mudar o objeto. **”**

12:11



13:41

**“** JavaScript é uma linguagem muito flexível. Flexível até demais, sob alguns aspectos. **”**



21:19

### Comandos

Em JavaScript, podemos definir variáveis de três formas, usando os seguintes comandos:



37:05

- var:** declara uma variável com escopo de função, podendo ter o valor de inicialização definido;
- let:** declara uma variável com escopo de bloco ou variável local, podendo ter o valor de inicialização definido;

## Objetos

Cada um dos objetos possui métodos e propriedades próprios. O JavaScript disponibiliza conjuntos diferentes de objetos, os quais são intrínsecos à linguagem utilizada, fornecidos pelo navegador (não fazendo parte do kernel da linguagem) ou fornecidos pelo ambiente que está sendo usado. Vale destacar que, quando rodamos como front-end, há um determinado conjunto de elementos que podemos usar, e quando rodamos como back-end, contamos com um outro conjunto de recursos.

## Tipos de dados

- a) **Undefined**: representa um valor de variáveis que não receberam nenhuma atribuição, que não foi inicializada e que, portanto, não pode sequer ter o seu tipo definido;
- b) **Null**: representa a ideia de “nada ou vazio”;
- c) **Boolean**: valores true ou false;
- d) **String**: valores de sequência imutável de caracteres, representada entre aspas duplas ou simples;
- e) **Number**: valores numéricos de ponto flutuante até 64 bits;
- f) **Object**: representa o conceito de objeto na linguagem;
- g) **Symbol**: representa identificadores únicos para objetos.

c) **const**: declara uma constante com escopo de bloco. A atribuição é única, mas possui valores de objeto mutável.

Nos casos em que a variável é classificada como “undefined”, sabemos que ainda não há uma definição estabelecida no código a respeito do seu tipo.

```
let a = m.split(" ")
switch(a[0]){
  case "connect":
    if(a[1]){
      if(clients.has(a[1])){
        ws.send("connected");
        ws.id = a[1];
      }else{
        ws.id = a[1]
        clients.set(a[1], {clients: position(ws.x, ws.y), id: ws.id});
        ws.send("connected")
      }
    }else{
      let id = Math.random().toString();
      id = id.substring(2, id.length - 4);
      clients.set(id, {clients: position(ws.x, ws.y), id: id});
      ws.id = id;
    }
}
```



41:16



43:20

“Lembra do nosso conceito de fracamente tipado? Eu criei uma variável, mas eu ainda não tenho ideia de como ela vai ser usada.”



47:47

## PALAVRA-CHAVE

**Math**: A biblioteca Math possui a definição de constantes e de operações matemáticas de uso geral. Assim, por exemplo, podemos representar os valores de  $\pi$  ( $Math.PI$ ), ou métodos relativos ao módulo ( $Math.abs()$ ), potência ( $Math.pow()$ ), entre outros.



50:12

## Arrays I

Um objeto do tipo Array permite o armazenamento de uma coleção formada por múltiplos itens, além de contar com propriedades e comportamentos específicos e executar operações comuns em arrays. O comprimento do array é definido na propriedade `length`, podendo ser aumentado através da atribuição de um valor a uma posição de índice maior ao tamanho atual.

## EXERCÍCIO DE FIXAÇÃO

Qual tipo de dado representa a ideia de “nada ou vazio”?



Vale destacar que os arrays são esparsos, de modo que só reservam espaço para elementos definidos, o aumento do seu tamanho gera posições intermediárias com valores indefinidos e pode-se usar o push para inclusão na última posição livre.



## AULA 3 • PARTE 2

### Funções

Definidas com a palavra reservada *function*, podem possuir um nome (embora exista o conceito de “função anônima”), podem ter argumentos e retornar valores. A passagem de parâmetros acontece por valor, e o número de parâmetros passados não é verificado.

Uma linguagem como o JavaScript não é uma linguagem de programação funcional, mas permite manipular funções e permite o uso de técnicas de programação funcional. Além disso, as funções podem ser manipuladas tal como os valores, podendo atribuir a variáveis, passar a função como parâmetro para outra função e retornar uma função como valor de retorno de outra função.

00:24



### Arrays II

07:09



09:03



Ao trabalhar com arrays, podemos fazer uso de diferentes métodos, retornando uma string com os valores do vetor, retornando novos arrays a partir da concatenação dos anteriores, particionando um array e retornando um novo com a partição, sem alterar o original, entre outras funções e recursos.

“ Existe o conceito de função anônima, a qual me diz que não preciso ter um nome pra minha função. ”

11:05



“ O que diferencia em programação procedural a função de um procedimento é que o procedimento não retorna algo daquele bloco de código, enquanto a função retorna. ”

## PALAVRA-CHAVE

**ECMAScript:** Lançada em 1997 e criada por Brendan Eich, da Ecma International, é uma linguagem de programação de uso geral baseada em scripts. A ECMAScript5, também conhecida como ECMAScript 2009, foi a primeira grande revisão da linguagem.



16:39



25:05

## PALAVRA-CHAVE

**Closure:** Também chamada de “função de fechamento”, trata-se de uma função que se “lembra” do ambiente em que foi criada, e que permite associar dados do ambiente com uma função que trabalha esses dados.



*Em um objeto, a gente tem propriedades, e essas propriedades têm valores que estão armazenados ali de fato.*



## Funções de alta ordem

O propósito das funções de alta ordem é explorar funções de transformação sobre arrays com arrow functions.

Há diferentes métodos de Array a serem explorados, como `Array.some()`, que sinaliza se ao menos um dos elementos do array atende a uma regra; `Array.every()`, que sinaliza se todos os elementos do array atendem a regra; `Array.filter()`, que retorna um novo array com a lista de itens que atende a regra; `Array.foreach()`, que itera sobre cada um dos itens da lista aplicando alguma ação; `Array.reduce()`, que acumula itens do array conforme a regra; `Array.map()`, que permite transformar os elementos da lista.



28:03



28:20



34:57

## PALAVRA-CHAVE

**Arrow functions:** As “funções de seta” representam uma sintaxe mais curta quando comparadas a uma função. Há, no entanto, algumas restrições associadas às arrow functions, como o fato de que não podem ser usadas como construtoras de objetos.

## AULA 3 • PARTE 3



00:24

## Modularização

É extremamente conveniente dividir e organizar código em módulos. Um módulo é um agrupamento de código que provê funcionalidade para outros

- “ Quando a gente trabalha sozinho, é importante que a gente tenha uma regra pessoal, que a gente entenda que a gente consiga fazer coisas que não nos prendam muito no desenvolvimento. ”**
- 01:21
- “ Quando a gente tá trabalhando em grupo, e isso vai acontecer muitas vezes, a gente tem que entender o que que vai ser feito em grupo para que façamos um desenvolvimento que seja o melhor possível. ”**
- 01:35
- “ O ECMAScript 6 Module veio para flexibilizar a forma como a importação de recursos é realizada, e ele já é suportado pelo Node.js e vários outros recursos, além de garantir questões de dinamicidade na importação de bibliotecas. ”**
- 06:22
- ES6**
- Padrão nativo do JavaScript disponível a partir do ECMAScript 6, lançado em 2015, também é suportado pelo ambiente de execução do Node.js. Os módulos definem suas interfaces por meio da palavra-chave export, podendo exportar múltiplas funções, classes, let, const e var. Nesse caso, vale destacar que a vinculação de exportação default é tratada como elemento principal do módulo. Por outro lado, dependências para outros módulos são importadas via palavra-chave import.
- “ Como seres humanos, a gente tem alguns atributos, algumas propriedades: nós temos olhos, cor de pele, idade... ”**
- 07:21
- “ Toda vez que se fala em propriedade e em atributos, é uma sinalização do que se tem, da alguma coisa que pode ser associada a um recurso. ”**
- 16:49
- 24:34
- 25:36
- 26:00
- módulos utilizarem sua interface, e especifica outros módulos que ele utiliza. Alguns dos benefícios da modularização giram em torno da facilitação da organização e da distribuição de blocos de funções e objetos relacionados, da permissão da reutilização de código, e da disponibilização de um “espaço de nomes” para evitar o compartilhamento de variáveis globais.
- 
- CommonJS**
- Padrão utilizado por muitos pacotes disponibilizados via NPM, é suportado pelo ambiente de execução do Node.js. No que tange aos módulos que contêm as definições, as interfaces são estabelecidas via exports e module.exports. Também se utiliza exports para adicionar propriedades ao objeto criado automaticamente pelo sistema de módulos; além do module.exports para definir o próprio objeto a ser retornado.
- Estudos de caso**
- No primeiro exemplo apresentado, o professor discute a representação de um carro (classe), bem como o acesso às informações a respeito desse automóvel (propriedades). Nesse sentido, há regras de comportamento que são dadas a determinados atributos, removendo, por exemplo, o acesso do usuário a esses atributos. A manipulação do atributo se dá pela verificação do comportamento. No caso de JavaScript, o caráter privado dos atributos é representado por #.
- No segundo exemplo, a classe herdada CarrosComPlaca e a classe Locadora

**“** Um array, em JavaScript, me permite que eu faça a construção de  $n$  elementos de fato ali. **”**

## EXERCÍCIO DE FIXAÇÃO

Qual tipo de dado representa valores true ou false?

são trabalhadas a partir do conceito de composição. Nesse caso, as definições do carro, que estão em outro diretório, são importadas e reutilizadas.



## AULA 3 • PARTE 4

00:24



### Desestruturação

Trata-se de uma operação comum na linguagem. A principal ideia da desestruturação é “desempacotar” algo em vários “pedaços”. Ela é aplicável em módulos importados, arrays e objetos, entre outros recursos. No caso dos vetores, por exemplo, a operação de atribuição pode utilizar um modo de “desestruturação”, “desempacotando” um array em diferentes segmentos.

Objetos também podem ser desestruturados em suas partes pelo operador de atribuição. Nesse caso, a ordem não importa, uma vez que o mapeamento se dá pelo identificador.

**“** Se eu criei uma classe cujos atributos são privados, qual é o comportamento que eu tenho na desestruturação? Porque, afinal de contas, eu não deveria ter acesso a esse recurso, que é privado. **”**

## JSON

Lançado em 2002, o JavaScript Object Notation é um formato compacto de troca de dados simples e rápida entre sistemas. Independente de linguagens, esse formato textual é usado na serialização de dados e no retorno de serviços web REST.

O JSON é capaz de representar tipos primitivos (Strings, números, booleanos, null) tranquilamente, além de tipos

03:54

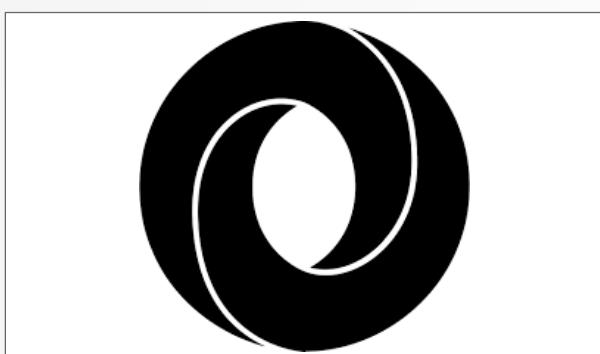


05:44



estruturados, como objetos (coleção não ordenada de zero ou mais pares chave-valor) e arranjos (coleção ordenada de zero ou mais valores).

No contexto de JavaScript, o método `JSON.stringify` converte um objeto para o formato JSON (String), mas é preciso estar atento para que não haja referências circulares no interior desse objeto. Já o método `JSON.parse` é capaz de converter uma String no formato JSON em um objeto. Essa conversão, vale destacar, não aceita qualquer tipo.



17:14



## Exceções

Falhas nas condições podem ser indicadas ao programador através do conceito de exceções. Quando uma função encontra uma situação anormal, ele informa tal anormalidade gerando uma exceção. Já o bloco de código, diante de uma situação anormal, captura a exceção e, em geral, indica que irá realizar o tratamento do problema identificado.

Para capturar e tratar exceções, utilizam-se os comandos `try`, `catch` e `finally`. No primeiro, estão colocados os comandos que podem provocar o lançamento de uma exceção. No segundo, as exceções são efetivamente capturadas. No terceiro, enfim, está o código a ser executado, independentemente da ocorrência de exceções.

17:40



21:53

23:30



24:00



**A exceção é um recurso muito interessante em toda e qualquer linguagem, pois ela é uma forma de tentar suavizar problemas que levariam à pausa abrupta do programa.**



## Funções assíncronas

A API de programação da linguagem JavaScript possui muitas funções de execução assíncrona. O pacote “fs” do Node.js, por exemplo, conta com diversas funções usadas na manipulação de arquivos de forma assíncrona.

Muitas APIs de JavaScript para funções assíncronas utilizam o conceito de funções de “call-back”, as quais são chamadas quando uma outra função encerrou seu processamento. Elas resultam em pequenas funções que são encadeadas para realizar um determinado processamento.

Além dos call-backs, as promises permitem o controle do fluxo de execução assíncrono de forma mais “limpa”. Elas representam o resultado, obtido pelo

**“** Se eu não tenho condições de tratar um erro, eu relanço o problema, a exceção, esperando que alguém, em outro momento, em outra parte do meu código, consiga fazer o tratamento adequado da exceção. **”**

**“** O principal foco [das funções assíncronas] é garantir que, independentemente do tempo de operação de uma determinada meta, eu vou seguir executando. **”**

## PALAVRA-CHAVE



32:34

**Async/await:** Async marca uma função ou método como sendo assíncrono. A palavra-chave await, por sua vez, quando usada antes de uma expressão que fornece um objeto Promise, faz com que o código espere até que a promise seja resolvida ou rejeitada. Async/await representam um modelo sintático para facilitar o uso de objetos Promise.

método then, ou a falha de uma operação assíncrona. Quando a promise for resolvida e assumir o estado “fulfilled”, a ação de um objeto promise terminou com sucesso; quando for rejeitada e assumir o estado “rejected”, diz-se que a promise terminou com falha. Vale destacar que as promises também podem ser encadeadas, sequencializando chamadas de funções assíncronas.



# Resumo da disciplina

Nesta página, veja um resumo dos principais conceitos trabalhados ao longo da disciplina.

## AULA 1

A Programação Orientada a Objetos (POO) é um paradigma de programação com foco em objetos, ao invés de funções.



Um objeto pode ser definido como uma coleção de dados e/ou funcionalidades com alguma relação entre si.

Em comparação ao modelo da programação estruturada, a POO nos permite agragar dados em objetos, de formas que as variáveis e as funções façam parte desses objetos.



## AULA 2

Os atributos não estáticos são aqueles que são parte do objeto. Em contraponto, os atributos estáticos são compartilhados por todas as instâncias de objeto.



As funções construtoras criam objetos de forma mais alinhada com a orientação objeto.



Trabalhar com orientação objeto é, também, trabalhar com relações entre objetos distintos.

## AULA 3

Definidas com a palavra reservada `function`, as funções podem possuir um nome, podem ter argumentos e retornar valores.



A principal ideia da desestruturação é “desempacotar” algo em vários “pedaços”.

Lançado em 2002, o JavaScript Object Notation é um formato compacto de troca de dados simples e rápida entre sistemas.



# Avaliação

Veja as instruções para realizar a avaliação da disciplina.

Já está disponível o teste online da disciplina. O prazo para realização é de **dois meses a partir da data de lançamento das aulas**.

Lembre-se que cada disciplina possui uma avaliação online.  
A nota mínima para aprovação é 6.

Fique tranquilo! Caso você perca o prazo do teste online, ficará aberto o teste de recuperação, que pode ser realizado até o final do seu curso. A única diferença é que a nota máxima atribuída na recuperação é 8.

